

Master Thesis Proceedings

Promotion 2023-2025

IFRoS



ELTE
EÖTVÖS LORÁND
UNIVERSITY

Erasmus Mundus Joint Master in
INTELLIGENT FIELD ROBOTIC SYSTEMS



With the support of the
Erasmus+ Programme
of the European Union

Contents

1	Perception-Aware Dynamic Locomotion via Attention-Guided World Model Distillation with Denoising Atharva Anil Patwe	4
2	TASK ALLOCATION AND STOP OPTIMIZATION FOR AUTONOMOUS WEED-ING david razhiel ceres arroyo	8
3	Robust LiDAR-Inertial Localization with Prior Maps in GNSS-Challenged Environments Eliyas Kidanemaraim Abraha	63
4	Indirect Visual-Inertial Simultaneous Localization and Mapping for Dense 3D Reconstruction Goitom Abrha Leaku	132
5	LiDAR Odometry and Mapping Beyond RTK Accuracy Liviu-Daniel Florescu	207
6	DEEP REINFORCEMENT LEARNING FOR DRONE OBSTACLE AVOIDANCE Thanh Loc Pham	276
7	Deep Reinforcement Learning for Robot Manipulation Vania Katherine Mulia	337
8	Sim2Real Transfer Learning for Object Tracking in Robotic Applications Mir Mohibullah Sazid	386
9	Fire Detection and Ranging forAutonomous Firefighting Drones Mohamed Khaled OthmanJune	461
10	ENHANCING INDOOR MAPPING AND LOCALIZATION IN SPECULAR RICH ENVIRONMENTS USING DEEP LEARNING AND SENSOR FUSION Renatto Tommasi Hernández	529
11	Detection of Invisible Obstacles for Drone Obstacle Avoidance Sawera Yaseen	588
12	Urban Object Detection Using Sensor Fusion for Autonomous Navigation Selin Yavuz	659
13	Enhancing Deep Reinforcement Learning with Curriculum Learning for Robotic Tasks Syma Afsha	716
14	Supervised Learning for Robot Manipulation Tanakrit Lertcompeesin	800
15	Vision-Based Tracking and Following of a Moving Target Using an Unmanned Aerial Vehicle Fatima Yousif Rustamani	863
16	Globally Consistent Mapping in Indoor and Outdoor Environments Using Hybrid LiDAR SLAM Zewdie Habtie Sisay	918

17	Joint Underwater Mapping with Acoustic and Optical Images	
	Precious Philip-Ifabiyi	990
18	Diurnal and Nocturnal Robust Visual-Aided GNSS Navigation on Horticultural Fields	
	Lisa Paul Magoti	1052

1

**Perception-Aware Dynamic Locomotion via Attention-Guided
World Model Distillation with Denoising**
Atharva Anil Patwe



Perception-Aware Dynamic Locomotion via Attention-Guided World Model Distillation with Denoising

Atharva Anil Patwe

Erasmus Mundus Joint Master's Degree

Supervisors:

Prof. Matko Orsag (*University of Zagreb*)

Prof. Stjepan Bogdan (*University of Zagreb*)

Tobias Jacob (*Neura Robotics GmbH*)

A thesis submitted to the Faculty of Electrical Engineering and Computing in partial fulfillment of the requirements for the Erasmus Mundus Joint Master's Degree in

Intelligent Field Robotic Systems

at the

University of Zagreb

· June 2025 ·

Abstract

Despite significant advances in humanoid robotics, achieving robust, adaptive locomotion in complex, real-world environments remains a formidable challenge, primarily due to the persistent gap between simulation and reality. This thesis confronts this challenge by developing and validating a perception-aware reinforcement learning framework that enables effective sim-to-real transfer for dynamic humanoid locomotion. The research utilizes the advanced 32-DOF 4NE-1 Gen3 humanoid robot as the primary development and evaluation platform.

This thesis introduces a three-phase reinforcement learning workflow for robust humanoid locomotion on the 4NE-1 robot. The first phase involves training an Oracle (teacher) policy in the high-fidelity simulation environment. This policy leverages privileged information, such as ground-truth contact forces, state, and terrain information, to achieve optimal performance, establishing a performance baseline. In the second phase, the Oracle's expert knowledge is distilled into a student policy. This is achieved through a dual process of training an attention-guided world model to denoise sensor data and infer latent states, and training the student policy to operate on realistic, on-board sensor inputs. The final phase consists of deploying the student policy on the physical 4NE-1 robot, with the potential for fine-tuning to bridge any remaining sim-to-real gap.

Extensive experiments demonstrate that the resulting policy generates stable, natural, and efficient bipedal locomotion. The robot successfully performs omnidirectional walking on flat terrain, navigates challenging rough terrain, and executes practical object transportation tasks. Quantitative analysis confirms high-fidelity velocity tracking, robust torso stability with minimal orientation error (<5 degrees), and remarkable resilience to external disturbances up to 150N. The findings illustrate a crucial trade-off where extensive domain randomization, while slightly increasing tracking error compared to the oracle, endows the student policy with the critical robustness required for real-world operation. This work presents a significant step toward deploying learned, perception-aware control policies on complex humanoid robots, with broad implications for their application in industrial automation, disaster response, and domestic assistance.

Humanoids are not about replacing humans, but about exploring the nature of being human...

Lay Abstract

Imagine teaching a robot to walk. A common way to do this safely is inside a computer simulation, like a video game. In this perfect digital world, the robot can learn quickly without breaking anything. However, the real world is far more complex and unpredictable, with uneven surfaces, unexpected bumps, and slippery spots. When a robot trained only in simulation tries to walk in the real world, it often stumbles and fails. This is known as the “reality gap.”

This research introduces a new method to help a humanoid robot, the 4NE-1, bridge this reality gap. Our approach works in three steps. First, we train an “expert” version of the robot in the perfect computer simulation, where it has access to all kinds of helpful information that isn’t available in the real world. This expert becomes a “teacher.”

Next, we train a “student” version of the robot’s brain. This student only has access to the information it would get from its own sensors in the real world, like cameras for seeing and balance sensors for stability. The teacher guides the student, teaching it how to make sense of noisy, incomplete information and how to walk effectively. A key part of this is an “attention” system that helps the student focus on the most important sensory details, much like how a person pays attention to specific things while walking.

Finally, we transfer this well-trained student brain into the physical 4NE-1 robot. The results are very successful. The robot can walk with a natural, stable gait, turn, and sidestep. It can navigate uneven ground and carry objects without falling. We even tested its balance by giving it strong pushes, and it remained steady. This work represents a significant step toward creating robots that can learn in the safety of simulation and then reliably bring their skills into our world to help with tasks at home, in factories, or even in disaster zones.

If you can't explain it to a six-year-old, you don't understand it yourself...

Albert Einstein

2

**TASK ALLOCATION AND STOP OPTIMIZATION FOR
AUTONOMOUS WEEDING**

david razhiel ceres arroyo

TASK ALLOCATION AND STOP OPTIMIZATION FOR AUTONOMOUS WEEDING

DAVID RAZHIEL CERES ARROYO



A Multi-Tool Allocation Approach for Optimized Weed Removal in Autonomous
Agriculture

June 2025 – David 145

David Razhiel Ceres Arroyo: *Task Allocation and Stop Optimization for Autonomous Weeding*, A Multi-Tool Allocation Approach for Optimized Weed Removal in Autonomous Agriculture, © June 2025

ABSTRACT

Hand weeding has traditionally been a labor-intensive and time-consuming task, making it an ideal candidate for automation. However, fields with high weed density remain a challenge for autonomous systems. In such scenarios, the limited number of robots and their tooling capacities create bottlenecks, leading to extended mission times. Nuga is a more capable system proposed by Paltech to improve throughput efficiency and reduce total mission duration. These benefits, however, depend on effectively solving the task allocation problem with a focus on minimizing tool idle time. To address this, we propose three task allocation algorithms of different paradigms: graph search, optimization-based, and market-based. Our results demonstrate that these approaches can reduce total mission time by up to 22.8% in high-density scenarios and decrease tool idle time by as much as 94.9% in comparison with the baseline method.

*Don't wish it was easier wish you were better.
Don't wish for less problems wish for more skills.
Don't wish for less challenge wish for more wisdom.*

— Jim Rohn

ACKNOWLEDGMENTS

Primero que nada, quiero agradecer profundamente a mis padres, *David y Lulú*, por apoyarme incondicionalmente y guiar mi camino con sus valiosas enseñanzas. Gracias por siempre dar lo mejor de ustedes para ayudarme a ser una mejor persona.

I would like to sincerely thank the IFROS program for supporting my journey in robotics, giving me the unique opportunity to study across different parts of the world, meet passionate individuals, and turn a dream into reality. This has enriched my life with unforgettable experiences and made me grow both personally and professionally.

I am also grateful to all my professors and my supervisor for generously sharing their knowledge and enthusiasm for the field. Their guidance has nurtured my curiosity and deepened my love for robotics.

Lastly, I want to thank *Felix* and the entire Paltech team for the opportunity to collaborate closely and contribute to such an exciting project. I am truly grateful for all the knowledge, experience, and trust you have given me throughout this journey.

CONTENTS

1	INTRODUCTION	1
1.1	Background and Context	1
1.2	Problem Definition	2
1.3	Related Work	2
1.4	Proposed Solution	5
2	METHODOLOGY	7
2.1	Task Allocation and Optimization	7
2.1.1	Metrics	7
2.1.2	Heuristics	9
2.1.3	Graph Search	11
2.1.4	Optimization	15
2.1.5	Market-based	18
3	RESULTS	22
3.1	The Robot	22
3.2	Simulation	22
3.2.1	URDF	23
3.2.2	SDF	24
3.2.3	Gazebo Plugins	25
3.2.4	Joint Control	25
3.2.5	Localization	27
3.2.6	Weed Detection	27
3.2.7	Nuga Controller	28
3.3	Task Allocation	30
3.3.1	Simulation Setup	30
3.3.2	Algorithm Comparison	30
3.3.3	Computation Time	35
3.3.4	Future Work	36
4	CONCLUSIONS	38
A	APPENDIX	39
A.1	Configuration Files	39
A.2	Algorithms	40
	BIBLIOGRAPHY	42

LIST OF FIGURES

Figure 1.1	Task Allocation classification.	2
Figure 2.1	Problem Layout Dimentions	8
Figure 2.2	Mission Dashboard	8
Figure 2.3	Heuristic Algorithm	10
Figure 2.4	Suboptimal solution computed using Heuristic	11
Figure 2.5	Optimal solution	11
Figure 2.6	Node types and cost representation	12
Figure 2.7	Graph expansion process	14
Figure 2.8	Graph Search Solution	14
Figure 2.9	Cost example of invalid solutions	15
Figure 2.10	Mixed-Integer Programming Solution	18
Figure 2.11	Bidding Mechanism Example	18
Figure 2.12	Market-based Solution	21
Figure 3.1	Nuga Platform	22
Figure 3.2	Simulator Components	23
Figure 3.3	Robot definition using URDF	24
Figure 3.4	Weed Infestation Example	25
Figure 3.5	Implement Tool (IT) Workspace Layout	26
Figure 3.6	ROS Interface for back gantry control	26
Figure 3.7	Example of training and detection	28
Figure 3.8	Nuga Controller ROS Interface	28
Figure 3.9	RViz Visualization	29
Figure 3.10	Weed distribution and coverage path in an agricultural field	30
Figure 3.11	Simulation Example	31
Figure 3.12	Low density algorithms comparison	31
Figure 3.13	Medium density algorithms comparison	32
Figure 3.14	High density algorithms comparison	32
Figure 3.15	Graph Search algorithm performance	36
Figure 3.16	Graph Search vs. Optimization	36

LIST OF TABLES

Table 1.1	Task Allocation approaches (1/2)	3
Table 1.2	Task Allocation approaches (2/2)	4
Table 1.3	Comparison between Task Allocation approaches	5
Table 3.1	Low-density Simulation Results	33

Table 3.2	Medium-density Simulation Results	34
Table 3.3	High-density Simulation Results	34

LISTINGS

Listing A.1	Weed Infestation World config example	39
Listing A.2	ROS2 config example	39

ACRONYMS

TA	Task Allocation
SVCA	Shapley Value Clustering Algorithm
CBDTA	Consensus-Based Distributed Task Allocation Algorithm
VDKM	Voronoi Diagram-Based, K-Means Algorithm
CBBA	Consensus-Based Bundle Algorithm
SSI	Sequential Single Item
CBPAEA	Consensus Based Parallel Auction and Execution Algorithm
GRU	Gated Recurrent Units
MLP	Multi-Layer Perceptron
EDACAM	Encoder Decoder Architecture with Cross Attention Mechanism
GNN	Graph Neural Network
PSO	Particle Swarm Optimization Algorithm
MIQP	Mixed-Integer Quadratic Program
STADAA	Sequential Task Addition Distributed Assignment Algorithm
GA	Genetic Algorithm
COQP	Constraint Optimization as Quadratic Program
IT	Implement Tool
DOF	Degrees of Freedom
URDF	Unified Robot Description Format
SDF	Simulation Description Format
WS	Workspace
DFS	Depth-First Search

viii **ACRONYMS**

BFS Breadth-First Search

MIP Mixed-Integer Programming

1

INTRODUCTION

1.1 BACKGROUND AND CONTEXT

Autonomous systems are complex agents capable of carrying out operations without human intervention. They have become more capable thanks to technological advancements and increasingly integrated into society with recent remarkable progress in artificial intelligence (AI) techniques. According to Zhang in [1], current trends indicate that the development and adoption of such systems will continue to grow in the coming years.

The agricultural sector is one of the areas where the integration of autonomous technologies has great potential. These systems could significantly benefit farmers by making their work safer and less repetitive. Autonomous systems have already been used in alternative cropping methods such as precision agriculture. Nevertheless, traditional practices are still facing challenges that autonomous systems could perfectly address. Among these challenges, the proliferation of weeds in grass fields raises as a major concern for the livestock well-being for two main reasons. First, weeds compete with grass for resources, causing forage loss [2]. Second, some weed species pose a direct threat to livestock health. In particular, plants like *Herbstzeitlose* (*Colchicum autumnale*) have been identified as toxic and the cause of livestock poisoning [3].

Removing these plants is a task that organic farmers must perform manually, as EU regulations restrict the use of pesticides and prevent farmers from combating weed proliferation through chemical means. It is evident that this task, especially in large grass fields, is highly time-consuming and extremely repetitive, making it an ideal candidate for automation. In Germany, companies like Paltech have developed solutions to address this problem using autonomous wheeled robots. Their flagship robot is a differential-drive wheeled system equipped with various sensors for localization and weed detection, as well as an onboard mechanism for weed removal. Currently, if the weed removal process needs to be sped up, the only solution is to deploy a fleet of robots. While this is feasible, developing single units capable of holding more than one weed control mechanism seems like the natural next step in Paltech's solution.

2 INTRODUCTION

1.2 PROBLEM DEFINITION

The development of systems with more than one mechanism for weed removal comes with both hardware and software challenges. It is crucial to ensure that tools and system resources are used as efficiently as possible. Paltech wants to avoid having more capable units with unused tools, especially since the production and deployment of these improved systems are more costly. Therefore, reducing idle time is a top priority and the focus of this thesis.

Idle time refers to periods when resources, such as tool equipment, are not actively engaged in productive work. Reducing idle time in this context means minimizing the time tools remain unused and maximizing their productivity in weed removal. To achieve this, allocating detected plants to the correct tools is essential. In the literature, this process is known as Task Allocation (TA). Some technical challenges to consider during implementation include computational latency and multi-tool coordination. The TA algorithm and execution pipeline must be fast enough to process new detections and reassign tools in real time without causing delays, while also ensuring that multiple tool units operate efficiently without interference or redundancy.

1.3 RELATED WORK

In general, a TA system aims to achieve an efficient assignment of tasks to robots (or tools in this case) by considering various characteristics such as the robots' capabilities, task requirements, and system efficiency. This process of TA involves three important factors to be considered according to Umashankar in [4]. Robot/tool, environment and coordination as shown in Figure 1.1.

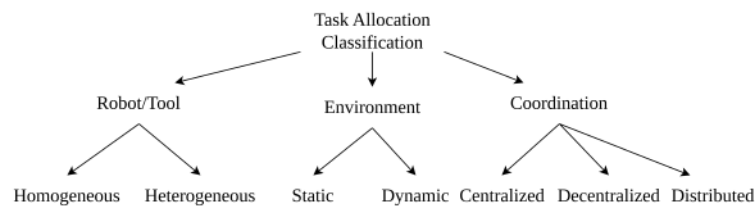


Figure 1.1: Task Allocation classification.

An example of homogeneous tools in this context is a robot equipped with multiple tools of the same type, such as drills. In contrast, heterogeneous tools refer to robots equipped with different types of tools, such as drilling, seeding, and sensing equipment.

The multi-tool TA can take place in either a dynamic or static environment. In an environment that is static in nature, tasks are allocated to tools in advance before they begin to execute them. This method works well in situations when the tasks are predetermined and the

environment remains unchanged. In contrast, dynamic TA involves the real-time assignment of tasks to tools as they carry out their activities. For the scope of this thesis, we will focus on homogeneous tools operating in a dynamic environment with centralized coordination, where the onboard computer will act as the master, assigning plants to each weed control mechanism.

There are several ways to accomplish multi-tool allocation, including heuristic, cluster-based, market-based, learning-based, and optimization-based techniques. Table 1.1 and 1.2 presents a comprehensive classification of TA algorithms found in the literature [4].

APPROACH	TECHNIQUE / ALGORITHM
Cluster Based	SVCA [5]
	Group Agent Partitioning [6]
	CBDTA [7]
	VDKM [8]
	CBBA [9]
	K-Means Clustering [10]
Market Based	Auction Algorithm [11]
	Improved Auction Algorithm [12]
	SSI Auction Algorithm [13]
	Extended SSI [14]
	Multihop-Based auction Algorithm [15]
	CBPAEA [16]
	Distributed Auction-Based Algorithm [17]

Table 1.1: Comparative overview of cluster-based and market-based approaches to Task Allocation.

In cluster-based approaches, the goal is to group tasks into a predefined number of clusters. Instead of assigning a single task to each tool, the clustering approach allocates entire groups of tasks to them, reducing the number of individual task assignments and computational complexity. Clustering approaches aim to minimize travel distance and maximize task coverage by grouping tasks effectively. However, the optimal clustering of tasks still needs further exploration. Although these approaches simplify TA, they struggle to handle dynamic changes in the environment.

An optimization-based strategy aims to select the best solution from a set of available options. These solutions are constrained by specific conditions, and the optimal one is determined based on the objective function. The objective function represents the system's ultimate goal. Some of the optimization algorithms have poor robustness

APPROACH	TECHNIQUE / ALGORITHM
Learning Based	GRU, MLP [18]
	Deep Reinforcement Learning [19]
	Heterogeneous Graph Attention Network [20]
	Capsule Attention-Based Mechanism [21]
	EDACAM [22]
	Graph Neural Network (GNN) [23]
Optimization Based	Mixed-Integer Quadratic Program [24]
	STADAA [25]
	PSO [26]
	Integer Programming [27]
	MIQP [28]
	Genetic Algorithm (GA) [29]
	COQP [30]
	Heuristic Based [31]
	Fuzzy Optimization [32]

Table 1.2: Comparative overview of learning-based and optimization-based approaches to Task Allocation.

to uncertainties therefore this approach is more suitable for solving well-defined and static problems focusing on theoretically optimal or near-optimal solutions. Additionally, optimization-based approaches require more computational power and are less adaptable to changing environments.

Market-based approaches effectively handle highly combinatorial optimization problems. In this method, an auctioneer informs agents about available tasks and requests bids. Each agent evaluates its capacity to complete the tasks and submits a bid accordingly. The auctioneer then assigns tasks to the agent with the most favorable bid. Generally, TA using this approach minimizes travel time. While these methods are flexible and scalable, they may not always achieve a globally optimal solution.

Recent approaches to TA incorporate deep learning techniques such as graph neural networks and graph convolutional networks. These types of TA methods are commonly referred as learning-based approaches. Most learning-based approaches struggle to generalize to larger-scale problem scenarios beyond those used during training. This characteristic is especially important because real-world TA problems frequently require modeling scenarios whose costs increase with the number of tasks and robots. Table 1.3 gives a comparison between all the approaches, Source [4].

	Clustering	Optimization	Market	Learning
Advantage	Simplifies TA and reduces complexity	Provides optimal solutions, suited for static problems	Flexible, scalable, decentralized	Adaptable, learns, and improves over time
Limitation	May not account for dynamics well	Computationally intensive, less adaptable	May not be global optima, needs effective bidding	Requires training, initially sub-optimal
Best case	Logical tasks	Well-defined, static problems	Dynamic environments with varying tasks	Complex and uncertain environments
Future work	Dynamic clustering, online adaptation	Hybrid models, real-time optimization	Adaptive market mechanisms, incentive models	Transfer learning, meta-learning

Table 1.3: Comparison between Task Allocation approaches

1.4 PROPOSED SOLUTION

As Table 1.3 illustrates, algorithm selection must be carefully considered based on the application's nature to achieve optimal performance. In a grass field clearing application, the environment is highly dynamic, especially since weed detections occur while the system is in motion. Therefore, market-based approaches are well-suited to ensuring the system adapts effectively to such conditions. However, this solution might not be optimal.

Optimization-based solutions, on the other hand, are known for providing optimal results, but they are typically more suited for static problems. In our case, although the environment is dynamic, this approach is still worth exploring. Since tasks are continuously added and removed from the allocation problem, the computation time at this task density may not be a significant issue, making real-time optimization feasible.

A graph search approach is also proposed, which becomes particularly interesting if the problem can be translated into a graph representation. Using well-known algorithms such as Depth-First Search (DFS) [33] or Dijkstra's [34] can help find the lowest-cost solution and achieve effective resource allocation, reducing both idle time and overall mission duration.

Lastly, clustering and learning-based approaches will be discarded. Clustering due to the challenges of dynamic clustering and the need for online adaptation. Although it offers the advantage of simplifying TA, its benefits diminish in low to medium density scenarios. Learning-based approaches are excluded because of the required training phase and the difficulty of generalizing across environments with varying task densities.

METHODOLOGY

2.1 TASK ALLOCATION AND OPTIMIZATION

Solving the [TA](#) problem in this context means assigning weed detections to removal tools in the most efficient way. The idea is to find the best allocation of resources and the optimal sequence of stops, that will maximize the number of plants removed, and minimize the tools' idle time. The solution has to consider the constraints of the problem such as:

GEOMETRIC CONSTRAINTS: A stop is considered valid only if the weeds lie within the workspaces of the tools (observe [Figure 2.1](#)).

MOVEMENT CONSTRAINTS: The robot can only move forward in a straight line at a max speed of $0.3 \frac{m}{s}$.

TASK PROCESSING: Each plant removal takes approximately 45 seconds, during which the robot must remain stationary to ensure a successful extraction. Movement during this process is not allowed as it could compromise the tool operation.

DYNAMIC ENVIRONMENT: Weed positions are initially unknown and are discovered dynamically during operation by the camera in front of the vehicle.

PROCESSING TIME: The solution must run online because the robot discovers new weeds during operation, and decisions about stopping and [TA](#) must adapt in real-time.

We define a 'good' stop as the next robot position that maximizes plant coverage while keeps a balance on the number of tasks assigned to each tool (aiming to minimize idle time). The problem can be approached in two ways: one option is to first determine the optimal stop location, after which [TA](#) is solved over the reachable tasks. Alternatively, we can first allocate tasks to the tools and then compute the stop position that satisfies the corresponding geometric constraints. Both approaches are similar, differing mainly in the order of operations, but each perspective opens the door to different methods and algorithms to try.

2.1.1 Metrics

Defining a good set of metrics is crucial to establish a solid basis for comparison between solutions and to easily identify the flaws of

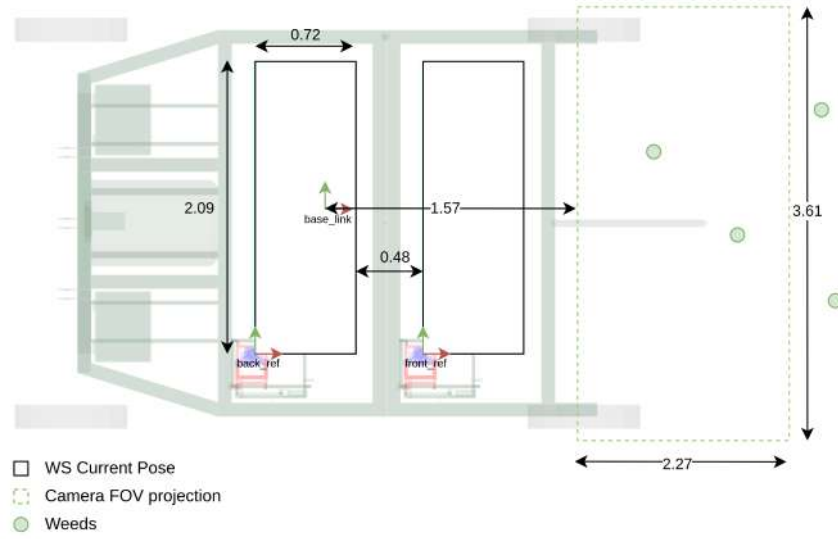


Figure 2.1: Problem Layout Dimentions

each method, thereby determining the best solution. To address this, we define two main categories. The first summarizes mission-level metrics such as the total idle and productive time of each tool, and the total time the robot spends moving or in a stationary state (defined by equations 2.1 and 2.2). The second category includes task-specific information, for example, task ID, status ('completed', 'failed', 'out'), number of stop, the tool that processed the task, and the idle and productive time of each tool at that specific stop. These metrics are displayed in a mission dashboard for easy visualization and comparison across missions using different scenarios or algorithms (see Figure 2.2b for the first category and 2.2a for the second one).

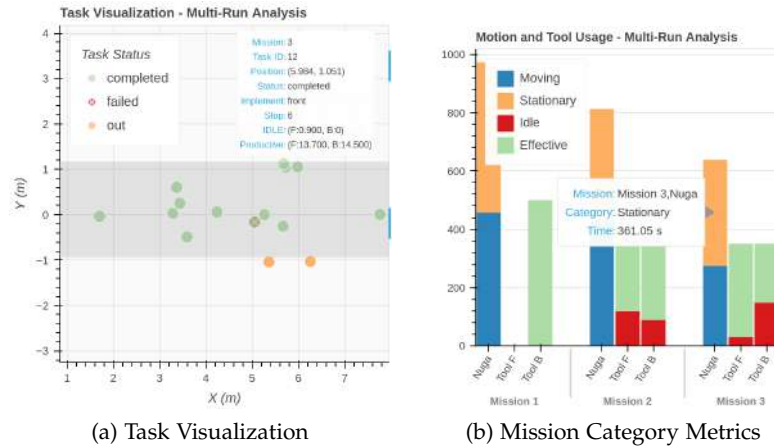


Figure 2.2: Mission Dashboard

FIGURE 2.2A Displays a 2D grid of the removed tasks' positions during the mission, along with their corresponding metadata. Hovering over a task reveals its details, and tasks can be filtered by mission.

FIGURE 2.2B Showcases a bar graph comparing different missions. Each mission includes the robot's moving and stationary time, as well as the idle and productive time of the onboard implement tools. A hover feature displays the value of each category.

$$t_{mission} = t_{moving} + t_{stationary} \quad (2.1)$$

$$t_{tool\_operation} = t_{idle} + t_{effective} \quad (2.2)$$

2.1.2 Heuristics

Heuristic solutions are commonly employed when the solution space of an optimization problem is too large to explore exhaustively, making an exact optimal solution computationally infeasible. They are also useful in time-sensitive scenarios, such as online implementations, where sacrificing optimality for efficiency is often justified.

In our work, we developed a heuristic algorithm to serve as a baseline solution that provides a meaningful reference point to assess the performance and potential improvements offered by other algorithms. The algorithm's description is detailed in 1, with an illustrative example in Figure 2.3.

Algorithm 1 Heuristic

- 1: Get the position of the closest weed from the current robot position.
 - 2: Project the tools' Workspace (WS) forward (in the future), aligning the trailing edge of the last WS with the closest weed's position plus a small clearance (e.g., 10 cm).
 - 3: Allocate weeds to each tool if they fall within its projected WS.
 - 4: Move the robot until the tools' WS are aligned with their projections, then execute the extractions for tools with assigned weeds.
 - 5: Repeat the process until mission has ended.
-

This approach offers a simple implementation and fast computation solution, making it well-suited for online applications. However, its heuristic nature leads to suboptimal solutions, as it does not account for minimizing idle time. In Figure 2.4 we exemplify the algorithm's suboptimality for a particular case, contrasting with a better solution to support our analysis.

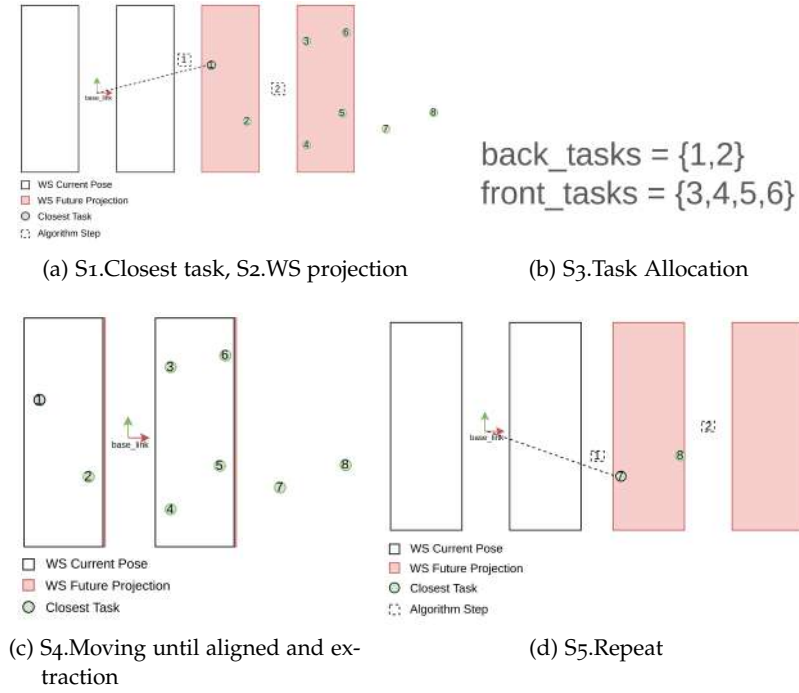


Figure 2.3: Heuristic Algorithm

FIGURE 2.4A Showcases the two steps required to remove eight detected weeds, with the first step allocating four tasks to the back tool and two to the front, the second step allocates remaining two tasks in the back.

FIGURE 2.4B Illustrates the idle and productive time for the given solution. Blue represents the robot's repositioning time, red indicates the idle time of the respective tool, and white the productive time. The first step shows a significant idle time for the front tool (90 seconds, equivalent to two tasks), caused by the tool waiting for the back tool to finish before proceeding to the next step. The second step shows a similar situation, where idle time arises from the only feasible option at that point: removing tasks 5 and 6 with the back tool.

An alternative solution for the same scenario (this time aimed at reducing idle time) is illustrated in [Figure 2.5](#). In this case, the solution requires three steps: first step assigns tasks 1 to back tool and task 5 to the front, the second step processes tasks 2 and 6, and finally the third step is used to remove tasks 3, 4, 7, and 8. As shown in [Figure 2.5b](#), the balanced TA between tools eliminates idle time and reduces the total mission duration compared to the previous solution.

This demonstrates the heuristic's suboptimality, as it fails to minimize idle time and does not consider the possibility of stopping at a location that allows for a more efficient TA. The heuristic algorithm is limited to the closest weed detection, which may not always be the best option.

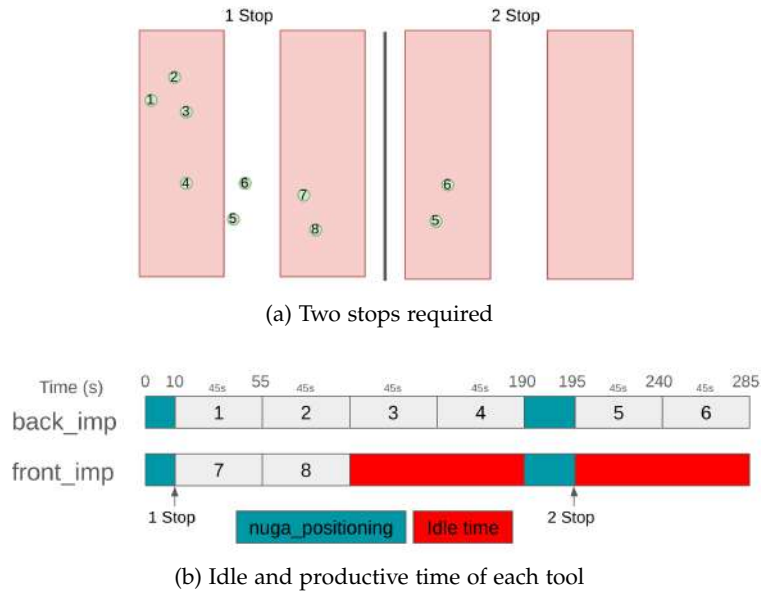


Figure 2.4: Suboptimal solution computed using Heuristic

In the following sections, we will explore more sophisticated algorithms that can overcome these limitations and provide better solutions.

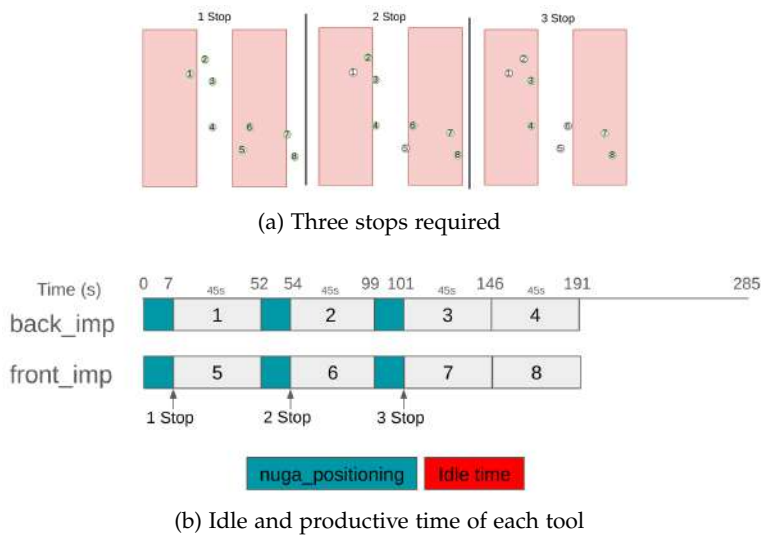


Figure 2.5: Optimal solution

2.1.3 Graph Search

Graph Search are a type of algorithms widely used in graph theory to systematically explore or traverse a graph. They are commonly used to find the shortest path between nodes, identify connected components, or solve various optimization problems. Graph search algorithms can

be broadly categorized into two main types: uninformed and informed search algorithms.

Uninformed search algorithms do not have any additional information about the problem domain and explore the graph blindly. Examples include DFS and Breadth-First Search (BFS). These algorithms are typically used for tasks like finding connected components or traversing all nodes in a graph. Informed search algorithms, on the other hand, use heuristics or additional information to guide the search process. They are often more efficient than uninformed algorithms for specific problems. Examples include A* search and Dijkstra's algorithm, which are commonly used for finding the shortest path in weighted graphs.

In our context, we exploit the advantages of graph-based algorithms by modeling the problem as such. Nodes represent potential stop locations, where two types of actions are possible: moving to another stop (a *stop node*) or performing weed removal at that location (an *action node*). Edge weights are defined as follows: edges between two *stop nodes* represent *travel cost*, computed using the distance between stops and the robot's velocity or a unitary value as a placeholder. Edges connecting a *stop node* to an *action node* represent *task execution cost*, and it is calculated based on the idle time (if any) of removing the indicated weeds with the assigned tools.

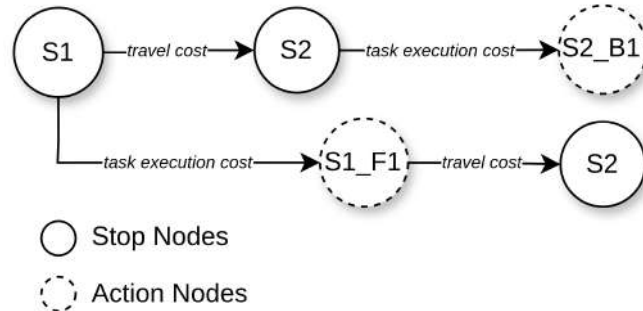


Figure 2.6: Node types and cost representation

FIGURE 2.6 Illustrates the convention used to represent the graph. S1 denotes stop number 1, while F or B indicate whether the front or back tool is assigned to remove the weed, followed by the weed ID to be processed. In this example, the task execution cost between S1 and S1_F1 would be 45s, as the back tool must wait for the front tool to process 1 plant. The same logic applies to the cost between S2 and S2_B1 but for the front tool.

The graph search implementation solution follows the next pipeline:

1. Compute **candidate stops** based on the robot's current position and weed detections. Candidate stops are the locations where

an important event occurs, such as a weed entering or exiting the workspace of a tool.

2. **Associate** reachable weeds with candidate stops. This allows the algorithm to determine which weeds can be removed at each stop.
3. Create a **graph representation** of the problem using **DFS** algorithm and the convention described in **Figure 2.6**. The graph is built by connecting *stop nodes* and *action nodes* using appropriate edge costs, taking into account the predefined associations between stops and tasks.
4. Get the **shortest path** in the graph from the root (first stop) to the final node (last stop) using Dijkstra's algorithm.
5. **Decode solution** and return the next optimal stop and **TA** as the algorithm's output.

To build and explore the graph, an implementation of the **DFS** algorithm is used (see Algorithm 4). The `get_children_nodes` method expands the graph by generating child nodes from a given parent node. If the parent node is a *stop node*, it creates child nodes for all valid combinations of reachable tasks that can be processed at that stop. Each combination forms a new child node that reflects the state after removing those tasks (*action node*). Additionally a 'trivial' *stop node* is added representing the decision to move to the next stop without removing any weeds. On the other hand, if the parent node is an *action node*, this method creates a single *stop node* representing the next stop. Each created edge is added to the graph with a weight based on the associated *task execution cost* or *travel cost*. **Figure 2.7** illustrates the graph expansion process, where the algorithm grows all possible combinations of tasks at each stop and generates the corresponding child nodes.

We observe that from S3 four possible states can be reached (three *action nodes* and one *stop node*). The transition S3_B1_F2 achieves zero idle time by optimally balancing tasks between the front and back tools, whereas the other transitions result in higher idle times due to imbalanced assignments. It's important to note that the same stop can be reached through different paths, depending on the actions taken at previous stops. For instance, if task 1 has been removed earlier, it should no longer be considered in subsequent stops. This consideration is crucial when building the graph, as we must ensure that it is expanded only through valid and realistic transitions that reflect feasible execution scenarios.

Having the complete graph, and given its configuration as a weighted directed graph, we choose Dijkstra's algorithm to find the optimal path. This algorithm efficiently explores the graph by maintaining a priority queue of nodes to be processed, ensuring that the node with

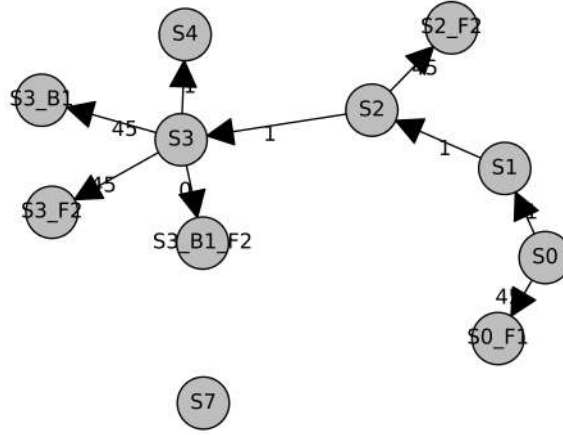


Figure 2.7: Graph expansion process

the lowest cost is always explored first. The algorithm continues until it reaches the final node, which represents the optimal solution. The path is then traced back from the final node to the root, revealing the sequence of stops and actions that lead to the optimal solution.

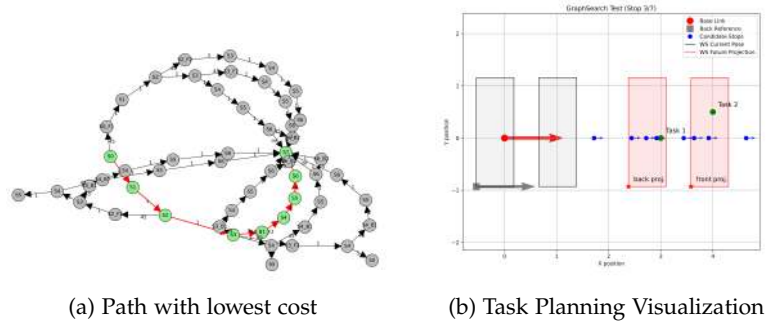
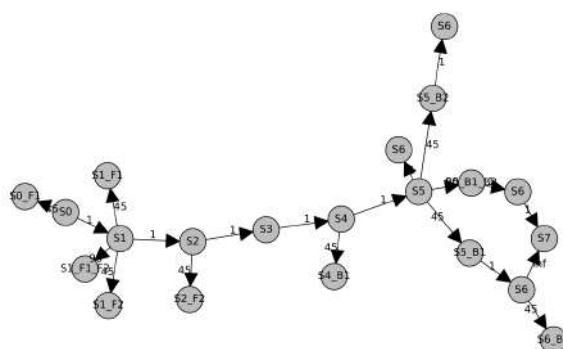


Figure 2.8: Graph Search Solution

An optimal solution obtained by the graph search algorithm in a scenario with two detected weeds is illustrated in Figure 2.8a. The highlighted branch corresponds to the optimal path found by the algorithm. For the given two weeds, the graph contained 54 nodes, and the computation time was 0.001 seconds. A visual representation of the solution is shown in Figure 2.8b, including the current robot pose, projected workspaces in red, candidate stops in blue, and weeds in green.

One remaining issue to address is preventing the algorithm from returning trivial solutions in which either no weeds are removed or not all of them are. To tackle this, each node keeps track of the tasks completed so far. If, at the second-to-last node, there are still tasks left to remove, we assign a penalty cost to the edge leading to the final node. This encourages the algorithm to prioritize removing all



Although the graph search algorithm is a promising approach, it still suffers from the exponential growth of the graph size as the number of weeds increases (as discussed in [Section 3.3.3](#)). This can lead to longer computation times and may not be suitable for real-time applications with high weed densities. In the next section, we will explore optimization-based approaches that can potentially overcome these limitations and provide more efficient solutions.

As discussed at the beginning of this chapter, the main goal of [TA](#) is to find the best allocation of resources and the optimal sequence of stops while considering the given constraints. Achieving such a solution requires mathematically formulating the scenario as an optimization task, with a clearly defined objective function, set of constraints, and decision variables that fully describe the problem. Optimization-based approaches are powerful tools that provide a systematic way to find the best feasible solution to a problem by exploring the solution space. Depending on the nature of the problem, a different optimization formulation can be used, such as linear programming, Mixed-Integer Programming ([MIP](#)), or nonlinear programming. In our case, we will focus on [MIP](#) as the most suitable approach for the problem.

MIP is a mathematical optimization technique that combines both integer and continuous variables in the formulation of the problem. It

allows for the modeling of complex decision-making scenarios where some variables must take on discrete values (e.g., binary decisions) while others can take on continuous values. This flexibility makes [MIP](#) particularly useful for problems that involve resource allocation, scheduling, and other combinatorial optimization tasks.

Given a set of tasks $\mathcal{T} = \{1, 2, \dots, T\}$ where each task $j \in \mathcal{T}$ corresponds to one weed detection from a total T , and a set of candidate stops $\mathcal{S} = \{1, 2, \dots, S\}$ where each stop $i \in \mathcal{S}$ is computed using events as in Graph Search algorithm. Let $x_{i,j} \in \{0, 1\}$ be a binary decision variable equal to 1 if task j is assigned to stop i , 0 otherwise and $imb_i \in \mathbb{N}_0$ the imbalance of assigned tasks between tools at stop i . We know that from each stop i we can only assign tasks that are visible from that stop, and a subset for those corresponds to tasks set to front or back tools, these statements could be better defined as:

$\mathcal{V}_i \subseteq \mathcal{T}$ Set of tasks visible from stop i

$\mathcal{F}_i \subseteq \mathcal{V}_i$ Front tool task assigned at stop i

$\mathcal{B}_i \subseteq \mathcal{V}_i$ Back tool task assigned at stop i

τ Constant processing time per task

We know that each task must be assigned to exactly one stop.

$$\sum_{i=1}^S x_{i,j} = 1 \quad \forall j \in \mathcal{T} \quad (2.3)$$

While visible tasks can only be assigned its corresponding stops.

$$x_{i,j} = 0 \quad \text{if } j \notin \mathcal{V}_i \quad (2.4)$$

The imbalance per stop is the absolute time difference between front and back tools.

$$f_i = \sum_{j \in \mathcal{F}_i} x_{i,j} \quad \text{and} \quad b_i = \sum_{j \in \mathcal{B}_i} x_{i,j} \quad (2.5)$$

We want the imbalance imb_i to reflect the time difference between front and back processing times at each stop. Therefore, we define the imbalance as the maximum of the two possible differences.

$$|\tau(f_i - b_i)| = \max(f_i - b_i, b_i - f_i) \quad (2.6)$$

Thus

$$imb_i \geq \tau(f_i - b_i) \quad \text{and} \quad imb_i \geq \tau(b_i - f_i) \quad (2.7)$$

Or equivalently

$$imb_i \geq |\tau(f_i - b_i)| \quad \forall i \in \mathcal{S} \quad (2.8)$$

The objective function is to minimize the total idle time, expressed as follows

$$\min \sum_{i=1}^S imb_i \quad (2.9)$$

Given the constraints and objective function, we can formulate the optimization problem as follows:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^S imb_i \\ & \text{subject to} && \sum_{i=1}^S x_{i,j} = 1 \quad \forall j \in \mathcal{T} \\ & && x_{i,j} = 0 \quad \text{if } j \notin \mathcal{V}_i \end{aligned} \quad (2.10)$$

An advantage of an optimization-based approach is that once the problem is formulated we can use existing optimization libraries to obtain the solution. In our case, we used the Google OR-Tools¹ library to solve our combinatorial problem.

The pipeline for the optimization-based approach is as follows:

1. Compute **candidate stops** based on the robot's current position and weed detections.
2. **Associate** reachable weeds with candidate stops.
3. Build the **optimization model** using decision variables, constraints, and objective function.
4. **Call solver** to get solution.
5. **Decode solution** and return the next optimal stop and **TA** as the algorithm's output.

This method does not suffer from high processing times as weed density increases. The algorithm consistently returns a solution in less than 0.1 seconds across a range of weed densities, from low to extremely high (tested up to 6.25 weeds/m²), making it a strong candidate for real-time applications. The optimization model effectively balances tasks between tools while minimizing idle time. [Figure 2.10](#)

¹ OR-Tools is an open source software suite for optimization, tuned for tackling the world's toughest problems in vehicle routing, flows, integer, linear, and constraint programming. <https://developers.google.com/optimization>

illustrates an example of the solution generated by the optimization algorithm in a scenario with twenty weeds.

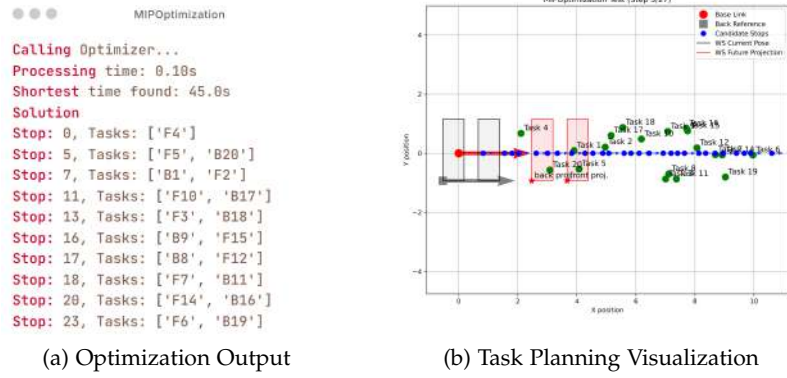


Figure 2.10: Mixed-Integer Programming Solution

2.1.5 Market-based

Market-based approaches are a class of algorithms that leverage the principles of supply and demand to solve TA problems. In our context, we consider the set of visible tasks from each candidate stop ($\mathcal{V}_i \subseteq \mathcal{T}$). Given these tasks, the algorithm forms all possible combinations of allocations and generates a bid for each one. The allocation is then determined based on the bids received and selecting the best one. A 'better bid' is defined as the combination that maximizes the number of removed weeds while minimizing the imbalance between tools (i.e., idle time). An illustrative example of the bidding mechanism is shown in Figure 2.11.

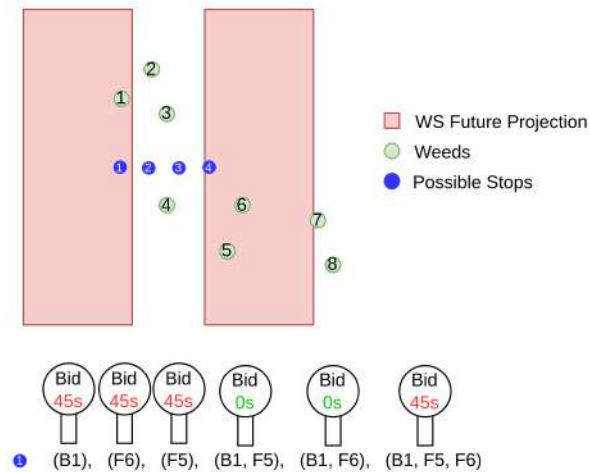


Figure 2.11: Bidding Mechanism Example

FIGURE 2.11 Illustrates a scenario with eight weed detections. The projected future workspace is marked in red and is positioned as if the robot were placed at candidate stop 1. Subsequent candidate stops are marked with blue circles. All possible TA combinations for the first stop are shown, each with its corresponding bid. The best bid is minimizing tool idle time while maximizing the number of removable tasks. This process is repeated for all stops, and the stop with the best overall bid is selected as the winner.

The algorithm is designed to be fast and efficient, making it suitable for real-time applications. Nevertheless, since the approach always selects the best next stop, it falls into the category of greedy algorithms due to its shortsighted nature. In some cases, it is necessary to sacrifice productivity at the next stop to benefit the overall solution, as illustrated in Figure 2.4 and Figure 2.5. The main advantage of this approach is its ability to adapt to dynamic environments where tasks change over time.

The next stop computation follows a similar pipeline to previous algorithms, with the main differences being the use of a bidding process to determine the best solution and the computation of candidate stops. Algorithm 2 outlines the steps involved during the algorithm.

Algorithm 2 Run Algorithm (Market-Based)

Require: Current tasks and robot pose

Ensure: Best stop and task assignment with shortest idle time

```

1: candidate_stops ← get_candidate_stops()
2: associate_tasks_with_stops(candidate_stops)
3: if log_print then
4:   log("Collecting Biddings...")
5: end if
6: start_timer()
7: (stop, tasks, idle_time) ← get_best_bid()
8: sw_time ← stop_timer()
9: if log_print then
10:  log("Processing time: " sw_time"s")
11:  log("Shortest time found: " idle_time"s")
12: end if
13: return (stop, tasks, idle_time)

```

The candidate stops are computed by considering only the closest weed detection and the events this task generate within the back tool workspace. These events include:

1. Entering the workspace.
2. The stop position where the weed is about to exit (rather than when it fully exits).

Using these two candidate poses, we generate evenly distributed positions to create additional candidates between these two events. Afterwards, we perform task association per stop as usual and call `get_best_bid` to start the bidding process and obtain the solution. The algorithm iterates through all possible combinations of tasks at each stop, calculating the idle time for each combination. The combination with the lowest idle time and more number of weeds is selected as the best bid. This process is repeated for all candidate stops, and the best overall stop and `TA` are returned (observe Algorithm 3).

Algorithm 3 Get Best Bid

Require: Candidate stops with associated tasks

Ensure: Best stop and task assignment with minimum idle time

```

1: bid  $\leftarrow$  Bid( $\emptyset$ ,  $\infty$ )
2: stops  $\leftarrow$  []
3: best_stop  $\leftarrow$  None
4: best_bid  $\leftarrow$  None
5: for  $i, \text{stop} \in \text{enumerate}(\text{candidate\_stops})$  do
6:   tasks_per_stop  $\leftarrow$  []
7:   for task  $\in$  stop["tasks"] do
8:     tasks_per_stop.append(task)
9:   end for
10:  combinations  $\leftarrow$  get_all_combinations(tasks_per_stop)
11:  idle  $\leftarrow$  get_idle_per_combination(combinations)
12:  bid  $\leftarrow$  get_bid(zip(combinations, idle))
13:  if update_bid_if_better(bid) then
14:    best_stop  $\leftarrow$   $i$ 
15:    best_bid  $\leftarrow$  bid
16:  end if
17: end for
18: return stops[best_stop], best_bid.tasks, best_bid.idle

```

An example of the solution produced by this algorithm is illustrated in Figure 2.12. We can observe that only four candidate stops are generated for four tasks. Notice the contrast with the graph search in Figure 2.8b and the optimization approach in Figure 2.10b.

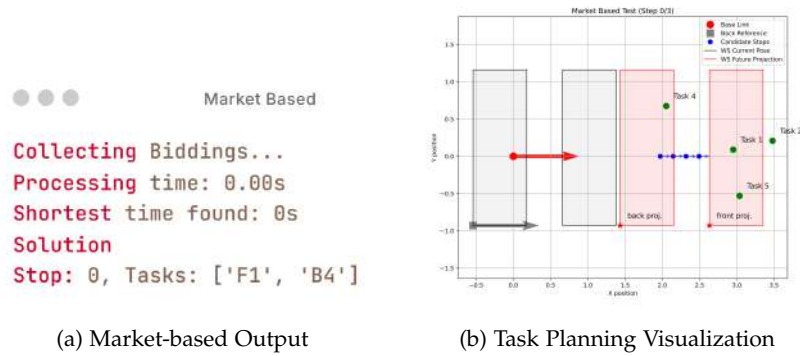


Figure 2.12: Market-based Solution

3

RESULTS

This chapter presents a detailed explanation of how the simulation was built and a performance comparison of all proposed **TA** methods across different scenarios ranging from low to high weed density. The goal is to identify the most effective solution to the **TA** problem and quantify the performance improvements over the baseline method. Additionally, this chapter clarifies key implementation details and evaluation metrics.

3.1 THE ROBOT

Nuga is Paltech's solution for speeding up the weed removal process. Nuga is a mobile platform equipped with two weed control mechanisms, also called **IT**, one main camera at the front for plant detection, two internal cameras for fine adjustment during tools' placement, an IMU, and two GNSS antennas for GPS localization. Each **IT** is mounted on a structure with three Degrees of Freedom (**DOF**) using prismatic joints, allowing movement in X, Y, and Z directions.

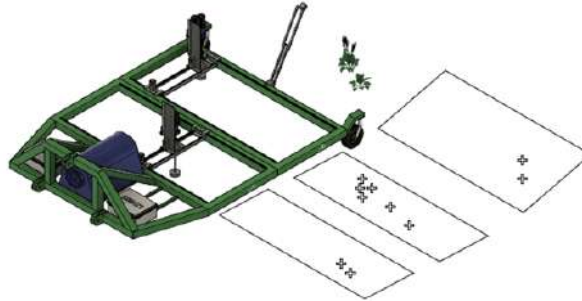


Figure 3.1: Nuga Platform

3.2 SIMULATION

A representative simulation of reality is crucial for developing new algorithms and analyzing robot behavior before real-world implementation. Therefore, building a simulation of the project was a foundational step for this work, ensuring a controlled environment for validation and testing. Gazebo¹ was the selected tool because it pro-

¹ Gazebo is a physics-based robotics simulation tool that allows testing and validation of robot models before real-world deployment. <https://gazebo.org/home>

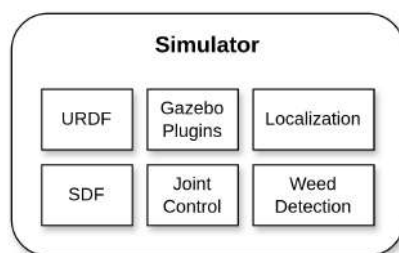


Figure 3.2: Simulator Components

vides a physics engine, supports sensor and actuator modeling, and integrates well with ROS², making it ideal for testing robotic systems.

The simulation consists of six key components: URDF files define the robot’s structure and properties, SDF files describe the virtual environment, and Gazebo plugins provide additional functionality, such as simulating custom sensors, actuators, or control interfaces. Additionally, core system operations include joint control for managing the movement of the *IT*, localization for tracking the robot’s position, and weed detection, which relies on an AI model for Rumex recognition. Figure 3.2 illustrates these components as building blocks for the simulation.

3.2.1 URDF

Unified Robot Description Format (URDF) is an XML file used to describe multibody systems for robot simulation. It defines the visual, collision, and inertial properties of rigid body objects, as well as their connections (*joints*). This establishes a spatial relationship between frames, which ROS and Gazebo can later interpret for control and visualization. This file also allows modeling different types of sensors and incorporating Gazebo plugins to link it with ROS control actions. We exploit these capabilities to define camera intrinsics, IMU behavior, GPS properties, and control the *IT* using `ros2_control`³.

Figure 3.3a displays the structure of the URDF files, being nuga the highest level entity that joins the robot description, gazebo sensor modeling and plugins, as well as `ros2_control` configuration. Nuga description defines the robot’s physical structure, including its links (e.g., chassis, wheels, camera support), joints (fixed, continuous, prismatic connections), sensors (cameras, IMU, GPS), and inertial properties. It organizes these components into a kinematic tree

² ROS (Robot Operating System) is an open-source framework that provides tools, libraries, and conventions for developing, managing, and simulating robotic applications. <https://www.ros.org/>

³ `ros2_control` is a ROS 2 framework that provides a standardized interface for managing hardware, enabling modular and reusable control systems for robots. https://control.ros.org/rolling/doc/getting_started/getting_started.html

(e.g., base_link -> chassis_link -> wheels/sensors) using macros for modularity, and resulting in the model displayed in Figure 3.3b.

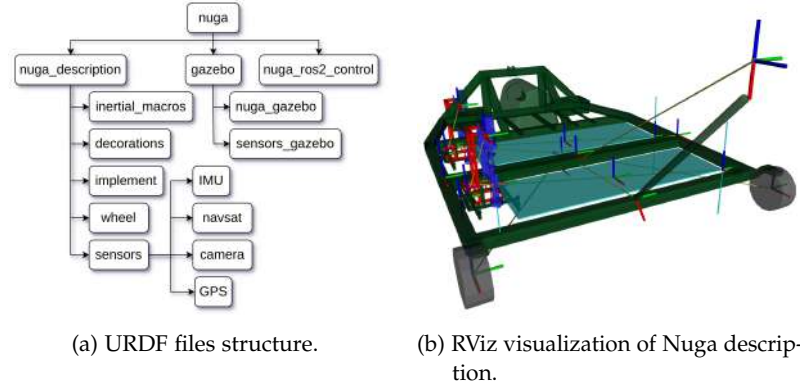


Figure 3.3: Robot definition using URDF

3.2.2 SDF

Simulation Description Format (SDF) also written in XML, describes the properties of the virtual world. Gazebo uses this file to define the terrain, obstacles, lighting conditions, physics parameters, and other environmental elements that affect the robot's interaction with the simulation. Having repeatability in a simulated world is important for debugging and testing purposes, for this reason a Python script was used to generate easy to configure worlds from a YAML configuration file. An example of the config file is shown in Listing A.1. For reproducibility, a seed value is configured in the simulation settings, the weed infestation pattern is defined within quadrants of specified dimensions (quadrant_size) and each quadrant is individually configured with:

- Spatial distribution:
 - *uniform*: Random uniform distribution
 - *clustered*: Random normal distribution with definable standard deviations (σ_x, σ_y)
- Weed density: Weeds per square meter (weeds/m²)
- Direction: Propagation axis for adjacent quadrants ($\pm x, \pm y$)
- Workspace expansion: If outside_workspace is true, the infestation area extends 10% beyond the quadrant boundaries.

A visual result of the generated world using such configuration file is shown in Figure 3.4.



Figure 3.4: Weed Infestation Example

3.2.3 Gazebo Plugins

The files `nuga_gazebo` and `sensors_gazebo` from [Figure 3.3a](#) instantiate and configure Gazebo plugins to define sensor behavior, including optical properties for the camera, as well as update rates and noise models for the IMU and GPS. The file `nuga_ros2_control` on the other hand, establishes an interface between the [IT](#)'s joints and `ros2_control` framework, specifying the command interface (position), controller type (forward position controller), and movement limits, enabling 3-DOF prismatic motion for each tool. Regarding movement control of the Nuga vehicle, the `ros_planar_move` plugin satisfied all control requirements given the platform's kinematic constraints, eliminating the need for additional configuration.

3.2.4 Joint Control

The control of both [IT](#) units was handled using the `ros2_control` framework (configuration example shown in [Listing A.2](#)), as previously described. This framework provides a seamless transition between simulation and real hardware control. In this context, the `forward_command_controller` was used, which is recommended for simulation because it bypasses PID computations by directly sending commands to simulated joints. These joints already track positions perfectly, without the disturbances or error correction needed in real-world scenarios. Simulators like Gazebo inherently handle ideal position tracking, making closed-loop control redundant. However, when switching to real hardware, replacing it with a `position_controller` is needed but straightforward thanks to the flexibility of the framework.

Nuga's workspace layout and dimensions are shown in [Figure 3.5](#). The gantry carrying the [IT](#) operates within a zone of 2.09 m in the Y direction, 0.72 m in X, and 0.26 m in Z. An extraction cycle begins with the gantry moving in X and Y to position the tool above the plant, followed by a downward movement in Z to lower the drill and

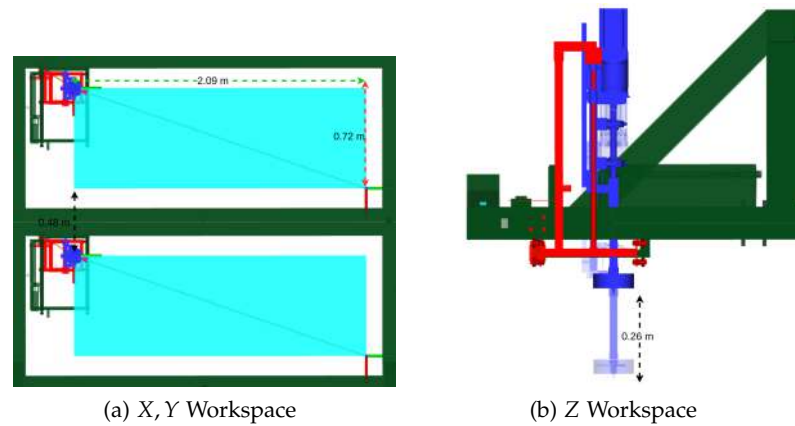


Figure 3.5: IT Workspace Layout

perform the extraction. The gantry has a maximum speed of $1 \frac{m}{s}$ in the XY plane, and each extraction can take up to 45 seconds per plant.

The IT is controlled using ROS2 actions, which provide a structured way to handle asynchronous tasks with feedback and result reporting. For the XY movement of the gantry, the `AxisPosition` action is used, allowing the specification of target coordinates (x, y) and speed, while providing feedback on the current position and confirming whether the target was reached. The `Extraction` action manages the vertical movement of the tool along the Z axis, reporting the depth reached, total time taken, and success status, along with real-time feedback on the current depth. Finally, the `ExtractionCycle` action coordinates the execution of multiple extractions by accepting an array of target poses and their corresponding IDs, providing feedback on the current status and reporting the results of the extraction process for each pose. These actions enable precise and modular control of the gantry system, ensuring efficient and reliable operation. A diagram summarizing this process is shown in Figure 3.6.

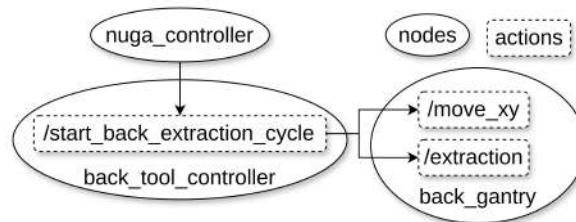


Figure 3.6: ROS Interface for back gantry control

3.2.5 Localization

Accurate localization is essential for mobile robot navigation. In most robotic systems, a combination of sensors such as wheel encoders, IMU's, and GPS (for outdoor applications) are used to estimate the robot's pose in the environment. In simulation, these sensors can be approximated to real-world conditions and provide a testbed for validating higher-level behaviours (e.g., path planning, task allocation, etc.). In this work, the `robot_localization`⁴ package was used to estimate the robot's pose by fusing sensor data. Typically, this would involve:

- An **Extended Kalman Filter (EKF)** fusing high-frequency wheel odometry and IMU data with low-frequency GPS.
- A `navsat_transform_node` to convert GPS data into the robot's odom frame.

However, as wheel odometry was not implemented in the simulation due to plugin limitations, the navigation was done merely using fusion of GPS and IMU, this allowed the EKF to still function and produce a pose estimate. While this approach deviates from standar practice, it was deemed acceptable for this thesis for several reasons: Localization is not the primary contribution of this work, the fused GPS and IMU data provided sufficient accuracy and stability for validating the [TA](#) investigation, and the use of `robot_localization` still maintains a realistic pipeline for future extension of real-world deployment.

This setup does not model short-term drift correction that would normally be provided by wheel odometry, and the reliance on GPS alone may introduce small jumps or inaccuracies in pose estimation. These limitations are acknowledged but do not affect the core objectives or validity of the experiments conducted in this work.

3.2.6 Weed Detection

Paltech uses an in-house machine learning model for Rumex recognition, trained on a dataset of images captured under various lighting conditions. The model is integrated into the simulation using a ROS2 node that subscribes to the camera topic and publishes detection results. This node processes camera images, applies the trained model, and outputs the detected plants' positions in camera coordinates along with confidence scores. This integration enables real-time detection of

⁴ `robot_localization` is a collection of state estimation nodes, each of which is an implementation of a nonlinear state estimator for robots moving in 3D space. It contains two state estimation nodes, `ekf_localization_node` and `ukf_localization_node`. In addition, it provides `navsat_transform_node`, which aids in the integration of GPS data. https://docs.ros.org/en/melodic/api/robot_localization/html/index.html

Rumex within the simulated environment, supporting planning and execution of extraction tasks. The model's training dataset includes images such as the one shown in Figure 3.7a, and it is capable of generalizing to simulation images, as illustrated in Figure 3.7b.

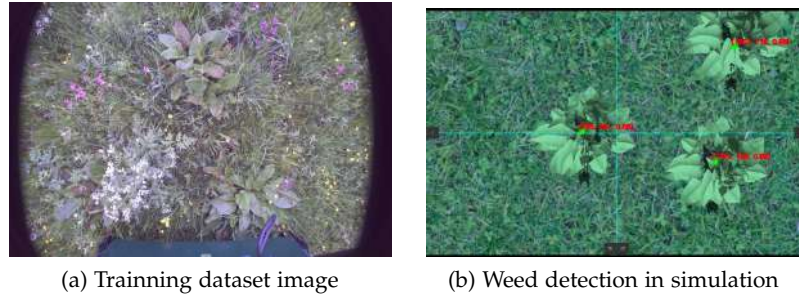


Figure 3.7: Example of training and detection

3.2.7 Nuga Controller

The Nuga controller is the main node responsible for orchestrating the movement of the robot, managing the **IT** units, and performing task allocation based on the weed positions received from the detection node. A high-level overview of the controller's operation is shown in Figure 3.8. The timer callbacks `get_ws_limits()` and `get_imp_offsets()` are self-destructive timers used to retrieve the positions of the tool workspace limits and their offsets relative to the base link. Once this information is received, the timers are destroyed. The same applies to the callback of the `/head_camera/camera_info` topic, which is used to obtain the camera's central pixel position for later use.

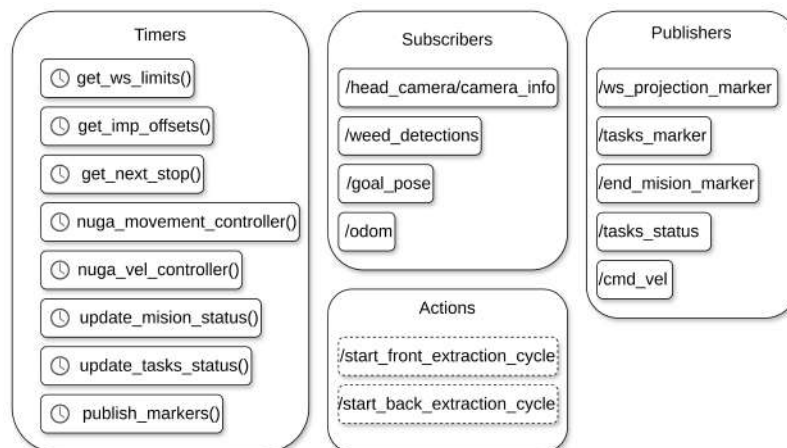


Figure 3.8: Nuga Controller ROS Interface

The `get_next_stop()` function is a timer callback that computes the robot's next stop when required. It uses the current detections, the

robot's pose, and the selected task allocation algorithm to determine the next target position, as well as the tasks to be executed at that stop by each IT. The execution steps followed by this method are shown in Algorithm 5.

Once a stop is determined, Nuga's movement and velocity controllers govern the robot's behavior. The `nuga_movement_controller()` function switches the robot's state between *stationary* and *moving* when needed, while also measuring the time spent in each state. If the robot reaches the target position, this method stops the robot and triggers the actions required to perform the extraction cycle; otherwise, it keeps the robot in the *moving* state. Finally, `nuga_velocity_controller()` takes the robot's state and computes the necessary velocity commands, publishing them to the `/cmd_vel` topic.

The callback `update_mission_status()` continuously checks whether the robot has reached the end-of-mission position in order to stop it and generate log information for mission metrics. On the other hand, `update_tasks_status()` continuously publishes the status of all weed detections to the `/tasks_status` topic. These tasks can include those assigned to the back or front tool, current detections, tasks outside the future path, and failed tasks.

Finally, `publish_markers()` updates RViz markers to visualize robot position and task status. Figure 3.9 shows an example of the RViz visualization in a usual mission.



Figure 3.9: RViz Visualization

FIGURE 3.9 Displays the current position of the robot (`base_link`), the tool workspaces, and tool positions using coordinate frames. Task categories are indicated with colored spheres: green for successful tasks, orange for tasks outside the path, red for failed tasks, and blue for robot stops. The workspace projections of the on-board tools are shown using red rectangles, while the green rectangle represents the front camera's field of view.

3.3 TASK ALLOCATION

Paltech currently offers its weeding solutions in fields ranging from 0.5 to 20 hectares, with an average weed density of 0.4 to 2.0 weeds/m². [Figure 3.10a](#) shows an example of usual weed density in a 1-hectare field, while [Figure 3.10b](#) illustrates the coverage path planning pattern that the robot follows in the same field. The simulations in this thesis are limited to the linear movements shown in [3.10b](#), due to the motion constraints described at the beginning of [Section 2.1](#). As a result, the simulation environments were designed accordingly.

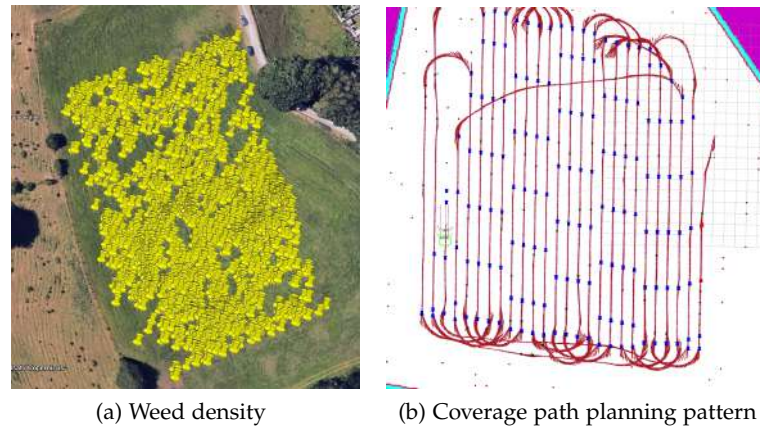


Figure 3.10: Weed distribution and coverage path in an agricultural field

3.3.1 Simulation Setup

Simulations were customized using YAML files, as mentioned in [Section 3.2](#), allowing for ease of configuration and logging convenience. All tests took place in a rectangular grass field of 10m × 50m, with different weed densities along a straight strip of 2m × 50m, simulating one line of the coverage path planning (observe an example in [Figure 3.11](#)). The local weed distribution within each quadrant (2m × 2m) varied in both type (*uniform* or *clustered*) and density, replicating irregular Rumex growth.

3.3.2 Algorithm Comparison

The first simulation corresponds to an infestation area of 100m², with a weed density of 0.28 weeds/m². This simulates a low weed density scenario, as shown in [Figure 3.12a](#). [Figure 3.12](#) presents both the task visualization and the mission metrics comparison across algorithms.

The first two missions in [Figure 3.12b](#) represent the baseline method using one and two tools, respectively. This comparison was made only to determine the natural improvement in mission time when adding

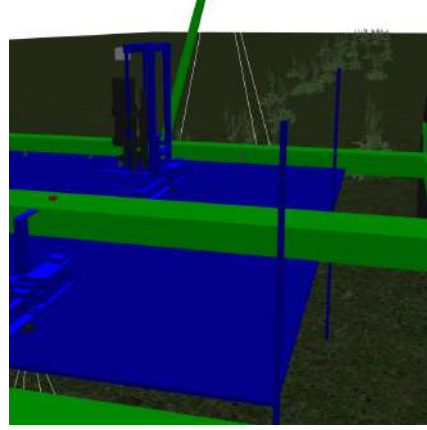


Figure 3.11: Simulation Example

an additional tool, even with a non-optimized algorithm. Missions three, four, and five correspond to the graph search, optimization, and market-based approaches, respectively. We observe similar mission times and no significant improvement over the baseline method, which is expected since the weed density is low and, most of the time, the separation between weeds prevents the robot from reaching them with both tools simultaneously. Nevertheless, we observe a reduction in tool idle time with algorithms three, four, and five. The heuristic algorithm results in an idle time of 12.8 min. for the front tool, whereas the others remain within the range of 8 min. to 9.8 min.

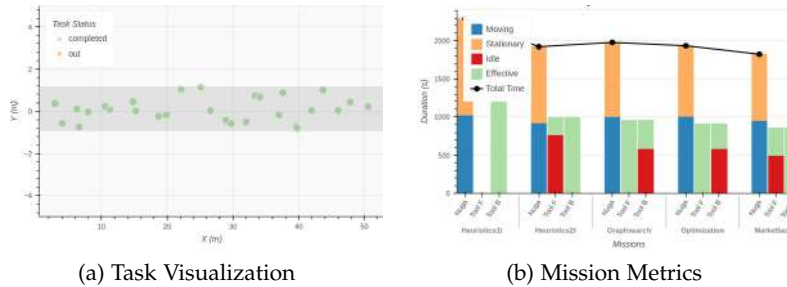


Figure 3.12: Low density algorithms comparison

For the second simulation, we increased the average weed density to 0.55 weeds/m², simulating a medium-density scenario, as shown in Figure 3.13a. In Figure 3.13b, we observe improvements in both mission time and tool productivity. Both the graph search and optimization-based algorithms show improvements compared to the heuristic approach. Specifically, while the baseline method overuses the back tool, resulting in high idle time for the front tool, graph search and optimization achieve more balanced tool usage. The market-based algorithm also reduces idle time, although not as significantly as the other two methods.

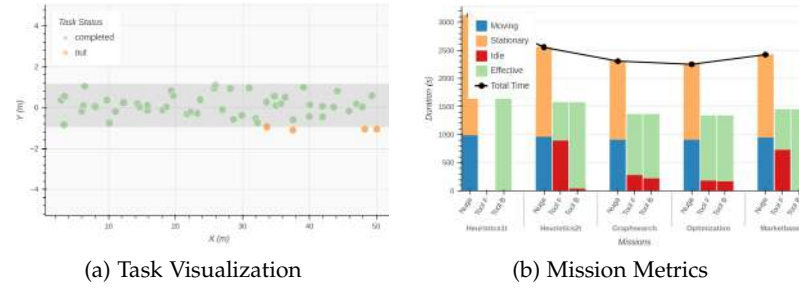


Figure 3.13: Medium density algorithms comparison

The third simulation corresponds to a weed density of 1.55 weeds/m^2 , simulating a high-density scenario, as shown in Figure 3.14a. Similar to the medium-density scenario (but more pronounced) the graph search and optimization-based approaches show better performance by reducing both mission time and idle time for both tools, achieving balanced tool usage. The market-based algorithm also shows noticeable improvements, achieving better tool productivity and mission time reduction than the baseline, although it still lags behind the other two methods.

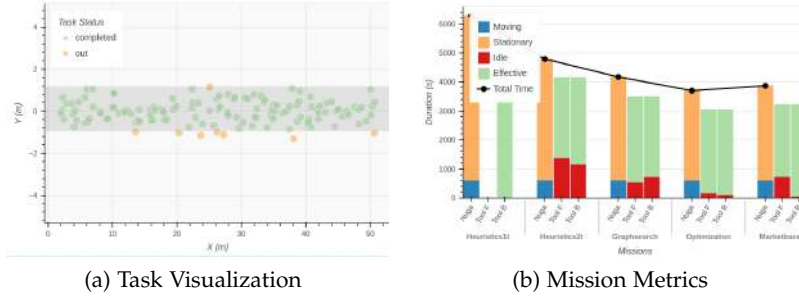


Figure 3.14: High density algorithms comparison

The simulation results are summarized in Table 3.1, Table 3.2, Table 3.3 for the low, medium, and high-density simulations, respectively. Each table presents the same information as the previous bar graphs under the 'Raw' column. Additionally, the 'Improvement' column shows the relative percentage difference of each algorithm compared to the baseline method for each category. This allows to determine the best algorithm for each weed density scenario.

		Raw (min)					Improvement (%)				
		Mov.	Stat.	Total	Idle	Eff.	Mov.	Stat.	Total	Idle	Eff.
Heuristics	N	15.3	16.7	32	-	-	-	-	-	-	-
	FT	-	-	-	12.8	3.7	-	-	-	-	-
	BT	-	-	-	0.1	16.4	-	-	-	-	-
Graph Search	N	16.7	16.2	32.9	-	-	9.2	-3.0	2.8	-	-
	FT	-	-	-	0.1	15.7	-	-	-	-99.2	324.3
	BT	-	-	-	9.8	6.1	-	-	-	9700.0	-62.8
Optimization	N	16.7	15.4	32.2	-	-	9.2	-7.8	0.6	-	-
	FT	-	-	-	0.1	14.9	-	-	-	-99.2	302.7
	BT	-	-	-	9.8	5.3	-	-	-	9700.0	-67.7
Market	N	15.8	14.5	30.3	-	-	3.3	-13.2	-5.3	-	-
	FT	-	-	-	8.3	5.9	-	-	-	-35.2	59.5
	BT	-	-	-	0.1	14.2	-	-	-	0.0	-13.4

Table 3.1: Low-density Simulation Results

TABLE 3.1 Confirms that there is no significant improvement in mission time nor tool productivity when using more sophisticated algorithms in low-density scenarios. We observe that the market-based approach was the only one to achieve a reduction in total mission time, with a 5.3% decrease compared to the baseline method. In contrast, the graph search and optimization methods actually increased the mission time by 2.8% and 0.6%, respectively.

Regarding tool productivity, we observe a shift in which tool experiences more idle time: from the front tool in the baseline to the back tool in the cases of graph search and optimization. Despite this shift, both methods reduced the back tool's idle time from 12.8 minutes to 9.8 minutes. The market-based method maintained the front tool as the one with the highest idle time but still achieved a 35.2% reduction.

		Raw (min)					Improvement (%)				
		Mov.	Stat.	Total	Idle	Eff.	Mov.	Stat.	Total	Idle	Eff.
Heuristics	N	16.1	26.4	42.5	-	-	-	-	-	-	-
	FT	-	-	-	15	11.1	-	-	-	-	-
	BT	-	-	-	0.7	25.3	-	-	-	-	-
Graph Search	N	15.2	23.1	38.4	-	-	-5.6	-12.5	-9.6	-	-
	FT	-	-	-	4.8	17.8	-	-	-	-68.0	60.4
	BT	-	-	-	3.8	18.8	-	-	-	442.9	-25.7
Optimization	N	16	24.5	40.6	-	-	-0.6	-7.2	-4.5	-	-
	FT	-	-	-	1.7	22.4	-	-	-	-88.7	101.8
	BT	-	-	-	9.8	14.2	-	-	-	1300.0	-43.9
Market	N	15.9	24.4	40.3	-	-	-1.2	-7.6	-5.2	-	-
	FT	-	-	-	12.2	11.8	-	-	-	-18.7	6.3
	BT	-	-	-	0	24	-	-	-	-100.0	-5.1

Table 3.2: Medium-density Simulation Results

		Raw (min)					Improvement (%)				
		Mov.	Stat.	Total	Idle	Eff.	Mov.	Stat.	Total	Idle	Eff.
Heuristics	N	10.3	69.4	79.8	-	-	-	-	-	-	-
	FT	-	-	-	23	46	-	-	-	-	-
	BT	-	-	-	19.5	49.6	-	-	-	-	-
Graph Search	N	10.3	59.1	69.4	-	-	0.0	-14.8	-13.0	-	-
	FT	-	-	-	9.1	48.9	-	-	-	-60.4	6.3
	BT	-	-	-	12.1	45.9	-	-	-	-37.9	-7.5
Optimization	N	10.3	51.2	61.6	-	-	0.0	-26.2	-22.8	-	-
	FT	-	-	-	2.8	47.8	-	-	-	-87.8	3.9
	BT	-	-	-	1.7	48.9	-	-	-	-91.3	-1.4
Market	N	10.3	54	64.4	-	-	0.0	-22.2	-19.3	-	-
	FT	-	-	-	12.2	41.4	-	-	-	-47.0	-10.0
	BT	-	-	-	1	52.6	-	-	-	-94.9	6.0

Table 3.3: High-density Simulation Results

TABLE 3.2 Highlights the improvements achieved in medium-density scenarios by the implemented algorithms over the heuristic approach. The graph search, optimization, and market-based methods reduced mission time by 9.6%, 4.5%, and 5.2%, respectively. Additionally, they achieved more balanced tool usage and reductions in idle time up to 88.7% for the front tool. The observed increment in back tool idle time is attributed to the heuristic method's unbalanced tool usage, which resulted in a very low idle time for only one tool, which is not desired.

TABLE 3.3 Demonstrates the effectiveness of the proposed algorithms in high-density scenarios. The graph search, optimization, and market-based methods achieved mission time reductions of 13.0%, 22.8%, and 19.3%, respectively. In terms of tool usage, all three methods achieved a balanced tool usage, with reductions in idle time of up to 87.8% for the front tool and up to 94.9% for the back tool.

3.3.3 Computation Time

The heuristic and market-based algorithms are considered short-sighted in nature due to the way they compute the next best stop without considering all tasks available up to that point. This characteristic makes their computation time negligible, regardless of the number of tasks currently available. In contrast, graph search and optimization-based algorithms take all available information into account when computing a solution, resulting in a larger solution space and, consequently, greater computational time required.

Our graph search solution performs well in terms of computation time, for a low to medium density of weeds. However, as the number of weeds increases, the graph size grows exponentially, leading to longer computation times. [Figure 3.15](#) showcases the graph size and computation time for different number of weeds (see plot in red). The graph size is defined as the number of nodes in the graph, while the computation time is the time required to build and find the optimal path in the graph.

We were able to optimize and reduce the graph size by applying a pruning technique that stops the growth of certain branches when they are not promising. This is achieved by introducing the concept of *expired tasks*, these are tasks that haven't been removed and can no longer be removed in future stops because the corresponding weeds have been left behind, outside the tool's workspaces. This pruning is implemented in the `get_children_nodes` method, where each task is checked for expiration before being added to the graph. As a result, the number of nodes is significantly reduced, improving computation time (observe green plot in [Figure 3.15](#)).

A similar processing time analysis was conducted for the optimization-based algorithm. Unlike the graph search approach, its computation

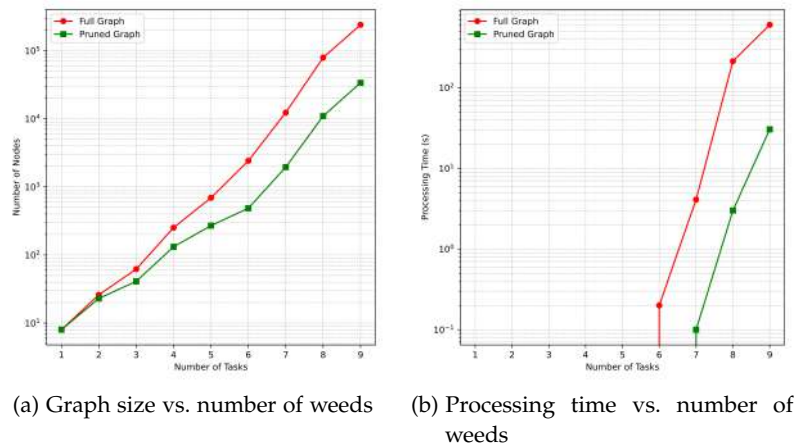


Figure 3.15: Graph Search algorithm performance

time scales well with the number of tasks and does not exhibit the same performance issues. This comparison is illustrated in Figure 3.16. Note that graph search is only a feasible solution for up to 10 tasks, beyond this point, the processing time becomes too high for practical use in online applications. In contrast, the optimization-based approach remains feasible for handling up to 300 tasks at a time.

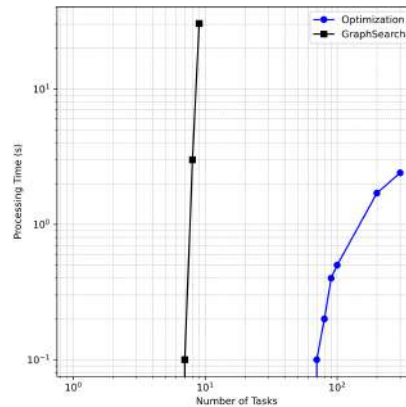


Figure 3.16: Graph Search vs. Optimization

3.3.4 Future Work

Observing improvements in mission duration and reduced idle time by adding an additional tool to the system and selecting the appropriate TA algorithm raises important questions such as: *To what extent does increasing the number of IT improve mission productivity? Is the cost-productivity trade-off justified when using more than two onboard tools?.* The simulation developed in this thesis lays the foundation for an-

swering such questions. By varying the number of IT and collecting comparative performance data, these analyses could be carried out if Paltech chooses to pursue them.

On another note, the dynamic nature of the task allocation problem means that information becomes available progressively as the mission takes place. Therefore, it is important to recognize that the solutions obtained are **locally optimal** based on currently available information. If the algorithm had access to all task information from the start, the likelihood of finding **globally optimal solutions** would be significantly higher. Questions such as *how much better those solutions would be*, and *what trade-offs they would involve*, remain open and are worth exploring. However, this scenario is hypothetical due to current hardware constraints, specifically, the limited field of view of the camera. The possibility of detecting tasks farther ahead, either through hardware changes or different sensing approaches, could provide the algorithm with more foresight, leading to better solutions. Investigating such enhancements is proposed as future work.

4

CONCLUSIONS

This thesis presents a comprehensive comparison between task allocation algorithms: heuristic, graph search, optimization-based, and market-based. Different paradigms were used to approach the problem of assigning detected weeds to onboard tools in the NUGA agricultural robotic platform. A simulation framework was developed to benchmark the performance of these algorithms across varying weed density scenarios, allowing for a detailed analysis of mission duration and tool idle time.

Results showed that both the graph search and optimization-based methods significantly outperform the baseline approach, particularly in medium to high-density scenarios. Notably, the optimization-based algorithm demonstrated strong scalability, maintaining acceptable computation times for larger problem sizes and reducing tool idle time by up to 94.9%. These findings highlight the potential of intelligent task allocation strategies to substantially improve the efficiency and productivity of autonomous field operations.

Investigating the impact of increasing the number of tools in the robot, analyzing productivity trade-offs and exploring globally optimal solutions by extending the robot's sensing capabilities are promising avenues for future research that could be explored using the current simulation.

This thesis demonstrates that intelligent task allocation strategies can significantly enhance the efficiency and productivity of field robotic systems, paving the way for more sustainable and scalable solutions in agriculture.



APPENDIX

A.1 CONFIGURATION FILES

Listing A.1: Weed Infestation World config example

```
seed: 2
quadrant_size: [2.0, 2.0]
quadrants:
  1:
    direction: x
    weed_density: 0.4 # weeds/m2
    spatial_distribution: clustered
    std_dev: [0.5, 0.5]
    outside_workspace: false
  2:
    direction: x
    weed_density: 1.2 # weeds/m2
    spatial_distribution: uniform
    outside_workspace: false
```

Listing A.2: ROS2 config example

```
# ROS2 Control
controller_manager:
  ros__parameters:
    use_sim_time: True
    update_rate: 20 # Hz

    joint_state_broadcaster:
      type: joint_state_broadcaster/JointStateBroadcaster

    forward_position_controller_front:
      type: forward_command_controller/ForwardCommandController
forward_position_controller_front:
  ros__parameters:
    joints:
      - front_x_axis_joint
      - front_y_axis_joint
      - front_implement_tool_joint
    interface_name: position
```

A.2 ALGORITHMS

Algorithm 4 Depth-First Search

Require: *root_node, last_node***Ensure:** Explored Graph

```

1: visited  $\leftarrow$  {last_node} {Initialize with last node}
2: stack  $\leftarrow$  [root_node] {Start with root node}
3:  $i \leftarrow 0$  {Iteration counter}
4: while stack is not empty do
5:    $i \leftarrow i + 1$ 
6:   node  $\leftarrow$  stack.pop()
7:   if node  $\notin$  visited then
8:     visited.add(node)
9:     children  $\leftarrow$  get_children_nodes(node)
10:    if children  $\neq$  None then
11:      stack.extend(children) {Add children to stack}
12:    end if
13:  end if
14:  if  $i == 5000$  then
15:    break {Early termination}
16:  end if
17: end while

```

Algorithm 5 Get Next Stop**Require:** Detections, Robot pose**Ensure:** Next stop pose and assigned tasks

```

1: if stop_pose  $\neq$  None then
2:   return
3: end if
4: if detections is empty then
5:   nuga_state  $\leftarrow$  "moving"
6:   return
7: end if
8: (tasks_in, tasks_out)  $\leftarrow$  get_tasks_in_future_path()
9: now  $\leftarrow$  current time
10: if tasks_out is not empty then
11:   for all (idx, pose)  $\in$  tasks_out do
12:     tasks_out_ws[idx]  $\leftarrow$  pose
13:     add_marker("map", now, "out", idx, pose, "orange")
14:   end for
15: end if
16: if tasks_in is empty then
17:   nuga_state  $\leftarrow$  "moving"
18:   return
19: end if
20: algorithm.set_robot_pose(nuga_pose)
21: algorithm.set_tasks(tasks_in)
22: (stop_pose, stop_tasks)  $\leftarrow$  algorithm.compute_next_stop()
23: add_marker("map", now, "goal", stops_counter + 1,
  stop_pose, "blue")
24: clear_polygons()
25: id  $\leftarrow$  0
26: for all (name, imp)  $\in$  implements do
27:   add_polygon("map", now, name + "_ws", id,
    imp.future_ws_pose)
28:   id  $\leftarrow$  id + 1
29: end for

```

BIBLIOGRAPHY

- [1] T. Zhang, Q. Li, Cs. Zhang, et al. "Current trends in the development of intelligent unmanned autonomous systems." In: *Frontiers in Information Technology & Electronic Engineering* 18 (2017), pp. 68–85. DOI: [10.1631/FITEE.1601650](https://doi.org/10.1631/FITEE.1601650).
- [2] Klötzli J., Suter M., Beaumont D. and Kolmanic A., Leskovsek R., and Schaffner U. "Key management practices to reduce the risk of the occurrence of *Rumex obtusifolius* in productive grasslands." In: *Weed Research* 1.64 (2021), pp. 76–88.
- [3] Mueller C., Sroka L., Hass M.L., Aboling S., These A., and Vervuert I. "Rejection behaviour of horses for hay contaminated with meadow saffron *Colchicum autumnale* L." In: *Journal of Animal Physiology and Animal Nutrition* 1.106 (2022), pp. 327–334.
- [4] Athira K A, Divya Udayan J, and Umashankar Subramaniam. "A Systematic Literature Review on Multi-Robot Task Allocation." In: *ACM Comput. Surv.* 57.3 (Nov. 2024). ISSN: 0360-0300. DOI: [10.1145/3700591](https://doi.org/10.1145/3700591). URL: <https://doi.org/10.1145/3700591>.
- [5] Javier G. Martin, Francisco Javier Muros, José María Maestre, and Eduardo F. Camacho. "Multi-robot task allocation clustering based on game theory." In: *Robotics and Autonomous Systems* 161 (2023), p. 104314. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2022.104314>. URL: <https://www.sciencedirect.com/science/article/pii/S0921889022002032>.
- [6] Junyan Hu, Parijat Bhowmick, Inmo Jang, Farshad Arvin, and Alexander Lanzon. "A Decentralized Cluster Formation Containment Framework for Multirobot Systems." In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1936–1955. DOI: [10.1109/TR0.2021.3071615](https://doi.org/10.1109/TR0.2021.3071615).
- [7] Xinye Chen, Ping Zhang, Fang Li, and Guanglong Du. "A cluster first strategy for distributed multi-robot task allocation problem with time constraints." In: *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*. 2018, pp. 102–107. DOI: [10.1109/WRC-SARA.2018.8584210](https://doi.org/10.1109/WRC-SARA.2018.8584210).
- [8] Jeongeun Kim and Hyoung Il Son. "A Voronoi Diagram-Based Workspace Partition for Weak Cooperation of Multi-Robot System in Orchard." In: *IEEE Access* 8 (2020), pp. 20676–20686. DOI: [10.1109/ACCESS.2020.2969449](https://doi.org/10.1109/ACCESS.2020.2969449).

- [9] Darren Smith, Jodie Wetherall, Stephen Woodhead, and Andrew Adekunle. "A Cluster-Based Approach to Consensus Based Distributed Task Allocation." In: *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. 2014, pp. 428–431. DOI: [10.1109/PDP.2014.87](https://doi.org/10.1109/PDP.2014.87).
- [10] Qi Lu, Joshua P. Hecker, and Melanie E. Moses. "Multiple-place swarm foraging with dynamic depots." In: *Autonomous Robots* 42.4 (2018), pp. 909–926. DOI: [10.1007/s10514-017-9693-2](https://doi.org/10.1007/s10514-017-9693-2). URL: <https://doi.org/10.1007/s10514-017-9693-2>.
- [11] Jiamei Lin, Pengcheng Li, Jialong Dai, Shaorui Liu, Wei Tian, Bo Li, and Changrui Wang. "Multi-robot Cooperative Assembly Task Planning for Large-scale Aerospace Structures." In: *2022 37th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. 2022, pp. 795–800. DOI: [10.1109/YAC57282.2022.10023897](https://doi.org/10.1109/YAC57282.2022.10023897).
- [12] Shiguang Wu, Xiaojie Liu, Xingwei Wang, Xiaolin Zhou, and Mingyang Sun. "Multi-robot Dynamic Task Allocation Based on Improved Auction Algorithm." In: *2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE)*. 2021, pp. 57–61. DOI: [10.1109/CACRE52464.2021.9501305](https://doi.org/10.1109/CACRE52464.2021.9501305).
- [13] Dong-Hyun Lee, Sheir Afgen Zaheer, Ji-Hyeong Han, Jong hwan Kim, and Eric Matson. "Competency adjustment and workload balancing framework in multirobot task allocation." In: *International Journal of Advanced Robotic Systems* 15.6 (2018), p. 1729881418812960. DOI: [10.1177/1729881418812960](https://doi.org/10.1177/1729881418812960). URL: <https://doi.org/10.1177/1729881418812960>.
- [14] Andong Shi, Shilei Cheng, Lei Sun, and Jingtai Liu. "Multi-robot Task Allocation for Airfield Pavement Detection Tasks." In: *2021 6th International Conference on Control, Robotics and Cybernetics (CRC)*. 2021, pp. 62–67. DOI: [10.1109/CRC52766.2021.9620140](https://doi.org/10.1109/CRC52766.2021.9620140).
- [15] Dong-Hyun Lee, Sheir Afgen Zaheer, and Jong-Hwan Kim. "A Resource-Oriented, Decentralized Auction Algorithm for Multirobot Task Allocation." In: *IEEE Transactions on Automation Science and Engineering* 12.4 (2015), pp. 1469–1481. DOI: [10.1109/TASE.2014.2361334](https://doi.org/10.1109/TASE.2014.2361334).
- [16] Gautham P. Das, Thomas M. McGinnity, Sonya A. Coleman, and Laxmidhar Behera. "A Distributed Task Allocation Algorithm for a Multi-Robot System in Healthcare Facilities." In: *Journal of Intelligent & Robotic Systems* 80.1 (2015), pp. 33–58. DOI: [10.1007/s10846-014-0154-2](https://doi.org/10.1007/s10846-014-0154-2). URL: <https://doi.org/10.1007/s10846-014-0154-2>.
- [17] Lingzhi Luo, Nilanjan Chakraborty, and Katia Sycara. "Distributed algorithm design for multi-robot task assignment with deadlines for tasks." In: *2013 IEEE International Conference on*

- Robotics and Automation*. 2013, pp. 3007–3013. DOI: [10.1109/ICRA.2013.6630994](https://doi.org/10.1109/ICRA.2013.6630994).
- [18] Xingjie Liu, Guolei Wang, and Ken Chen. “Option-Based Multi-Agent Reinforcement Learning for Painting With Multiple Large-Sized Robots.” In: *IEEE Transactions on Intelligent Transportation Systems* 23.9 (2022), pp. 15707–15715. DOI: [10.1109/TITS.2022.3145375](https://doi.org/10.1109/TITS.2022.3145375).
 - [19] Ahmed Elfakharany and Zool Hilmi Ismail. “End-to-End Deep Reinforcement Learning for Decentralized Task Allocation and Navigation for a Multi-Robot System.” In: *Applied Sciences* 11.7 (2021). ISSN: 2076-3417. DOI: [10.3390/app11072895](https://doi.org/10.3390/app11072895). URL: <https://www.mdpi.com/2076-3417/11/7/2895>.
 - [20] Zheyuan Wang, Chen Liu, and Matthew Gombolay. “Heterogeneous Graph Attention Networks for Scalable Multi-Robot Scheduling with Temporospacial Constraints.” In: *Autonomous Robots* 46.1 (2022), pp. 249–268. DOI: [10.1007/s10514-021-09997-2](https://doi.org/10.1007/s10514-021-09997-2). URL: <https://doi.org/10.1007/s10514-021-09997-2>.
 - [21] Steve Paul, Payam Ghassemi, and Souma Chowdhury. *Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks*. 2022. arXiv: [2205.03321](https://arxiv.org/abs/2205.03321) [cs.MA]. URL: <https://arxiv.org/abs/2205.03321>.
 - [22] Bumjin Park, Cheongwoong Kang, and Jaesik Choi. “Cooperative Multi-Robot Task Allocation with Reinforcement Learning.” In: *Applied Sciences* 12.1 (2022). ISSN: 2076-3417. DOI: [10.3390/app12010272](https://doi.org/10.3390/app12010272). URL: <https://www.mdpi.com/2076-3417/12/1/272>.
 - [23] Jacopo Banfi, Andrew Messing, Christopher Kroninger, Ethan Stump, Seth Hutchinson, and Nicholas Roy. “Hierarchical Planning for Heterogeneous Multi-Robot Routing Problems via Learned Subteam Performance.” In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4464–4471. DOI: [10.1109/LRA.2022.3148489](https://doi.org/10.1109/LRA.2022.3148489).
 - [24] Gennaro Notomista, Siddharth Mayya, Yousef Emam, Christopher Kroninger, Addison Bohannon, Seth Hutchinson, and Magnus Egerstedt. “A Resilient and Energy-Aware Task Allocation Framework for Heterogeneous Multirobot Systems.” In: *IEEE Transactions on Robotics* 38.1 (2022), pp. 159–179. DOI: [10.1109/TR0.2021.3102379](https://doi.org/10.1109/TR0.2021.3102379).
 - [25] Nicholas Lindsay, Robert K. Buehling, and Lihong Sun. “A Sequential Task Addition Distributed Assignment Algorithm for Multi-Robot Systems.” In: *Journal of Intelligent & Robotic Systems* 102 (2021), p. 51. DOI: [10.1007/s10846-021-01394-7](https://doi.org/10.1007/s10846-021-01394-7).

- [26] Zhi Li, Ali Vatankhah Barenji, Jiazhi Jiang, Ray Y. Zhong, and Gangyan Xu. "A Mechanism for Scheduling Multi Robot Intelligent Warehouse System Face with Dynamic Demand." In: *Journal of Intelligent Manufacturing* 31.2 (2020), pp. 469–480. DOI: [10.1007/s10845-018-1459-y](https://doi.org/10.1007/s10845-018-1459-y). URL: <https://doi.org/10.1007/s10845-018-1459-y>.
- [27] Shridhar Velhal, Suresh Sundaram, and Narasimhan Sundararajan. "Dynamic Resource Allocation With Decentralized Multi-Task Assignment Approach for Perimeter Defense Problem." In: *IEEE Transactions on Aerospace and Electronic Systems* 58.4 (2022), pp. 3313–3325. DOI: [10.1109/TAES.2022.3147742](https://doi.org/10.1109/TAES.2022.3147742).
- [28] Siddharth Mayya, Diego S. D'antonio, David Saldaña, and Vijay Kumar. *Resilient Task Allocation in Heterogeneous Multi-Robot Systems*. 2021. arXiv: [2009.04593](https://arxiv.org/abs/2009.04593) [cs.MA]. URL: <https://arxiv.org/abs/2009.04593>.
- [29] Kelin Jose and Dilip Kumar Pratihari. "Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods." In: *Robotics and Autonomous Systems* 80 (2016), pp. 34–42. ISSN: 0921-8890. DOI: <https://doi.org/10.1016/j.robot.2016.02.003>. URL: <https://www.sciencedirect.com/science/article/pii/S0921889016000282>.
- [30] Gennaro Notomista, Siddharth Mayya, Seth Hutchinson, and Magnus Egerstedt. *An Optimal Task Allocation Strategy for Heterogeneous Multi-Robot Systems*. 2019. arXiv: [1903.08641](https://arxiv.org/abs/1903.08641) [cs.R0]. URL: <https://arxiv.org/abs/1903.08641>.
- [31] Wanqing Zhao, Qinggang Meng, and Paul W. H. Chung. "A Heuristic Distributed Task Allocation Method for Multivehicle Multitask Problems and Its Application to Search and Rescue Scenario." In: *IEEE Transactions on Cybernetics* 46.4 (2016), pp. 902–915. DOI: [10.1109/TCYB.2015.2418052](https://doi.org/10.1109/TCYB.2015.2418052).
- [32] Oscar Valero, Javier Antich, Antoni Tauler-Rosselló, José Guerrero, Juan-José Miñana, and Alberto Ortiz. "Multi-robot task allocation methods: A fuzzy optimization approach." In: *Information Sciences* 648 (2023), p. 119508. ISSN: 0020-0255. DOI: <https://doi.org/10.1016/j.ins.2023.119508>. URL: <https://www.sciencedirect.com/science/article/pii/S0020025523010939>.
- [33] Robert Tarjan. "Depth-First Search and Linear Graph Algorithms." In: *SIAM Journal on Computing* 1.2 (1972), pp. 146–160. DOI: [10.1137/0201010](https://doi.org/10.1137/0201010). eprint: <https://doi.org/10.1137/0201010>. URL: <https://doi.org/10.1137/0201010>.

- [34] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs." In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. DOI: [10.1007/BF01386390](https://doi.org/10.1007/BF01386390). URL: <https://doi.org/10.1007/BF01386390>.

3

**Robust LiDAR-Inertial Localization with Prior Maps in
GNSS-Challenged Environments
Eliyas Kidanemaraim Abraha**



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTER
IN INTELLIGENT FIELD ROBOTIC SYSTEMS

Robust LiDAR-Inertial Localization with Prior Maps in GNSS-Challenged Environments

Author:

Eliyas Kidanemaraim Abraha
Intelligent Field Robotics Systems, MSc

Internal supervisor:

Zoltan Istenes
Associate Professor at ELTE

External supervisor:

Mohammad Aldibaja
Associate Professor at SMART

Budapest, 2025

Abstract

This thesis presents a real-time, map-based localization system for autonomous robots operating in large-scale, GNSS-denied, and partially dynamic environments. The system integrates LiDAR-Inertial Odometry (FAST-LIO2) with scan-to-map registration using the multithreaded Normal Distributions Transform (NDT-OMP) algorithm. These components are fused through a sliding-window factor graph optimization framework, enabling both high-frequency motion estimation and global drift correction.

To support scalability and computational efficiency, the system incorporates a dynamic submap loading strategy that retrieves only the local region of the prior 3D map relevant to the robot’s current pose. The use of NDT-OMP allows for parallel scan matching, contributing to real-time operation with a per-frame latency under 23 milliseconds. Additionally, a deep learning-based 3D object detection module (CenterPoint) is integrated to remove dynamic elements from the LiDAR input, improving registration stability in semi-dynamic scenes.

Experimental evaluations on both benchmark and custom datasets demonstrate that the system achieves centimeter- to decimeter-level localization accuracy while maintaining real-time performance. The system was further tested under simulated environmental degradation, including moderate fog noise, and maintained robust localization performance. It also proved effective in feature-sparse regions and unmapped transition zones, where scan-to-map alignment is typically unreliable. Finally, a modular architectural extension is proposed to support future integration of additional sensing modalities—such as vision or radar—enhancing adaptability in perceptually degraded environments. These contributions collectively advance the robustness, efficiency, and extensibility of real-time localization for autonomous systems.

Acknowledgements

I would like to express my sincere gratitude to my academic supervisor, **Dr. Zoltan Istenes**, and my industrial supervisor, **Dr. Mohammed Aldibaja**, for their continued guidance, encouragement, and generosity in sharing their knowledge throughout the course of my master's thesis. Their insights and support were instrumental in shaping the quality and direction of this work. I also extend my sincere appreciation to the **Smart Mechatronics group at Saxion University of Applied Sciences** for hosting my thesis and providing a supportive and engaging research environment.

I am deeply thankful to the professors at the **University of Girona** and **Eötvös Loránd University** for their dedicated instruction and support from the beginning of the **IFRoS program**. Their efforts have greatly contributed to my growth and understanding in the field of robotics. I also thank the **IFRoS program coordinators** for their exceptional organization and continuous assistance throughout the program.

I would like to thank all my **classmates and friends** in the IFRoS program for their companionship, teamwork, and encouragement during this enriching journey across countries and institutions.

On a personal note, I am deeply grateful to my **parents** and my two **wonderful sisters** for their unwavering love, constant motivation, and emotional support. Their belief in me has been a continuous source of strength and inspiration throughout this academic journey.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	iii
List of Algorithms	v
List of Abbreviations	vi
2 Introduction	1
2.1 Motivation	1
2.2 Objective	2
2.3 Research Questions	3
2.4 Scope and limitations	3
2.4.1 Scope	3
2.4.2 Limitations	4
2.5 Overview of the thesis structure	4
3 Litreature Review	6
3.1 Overview of Relevant Literature	6
3.1.1 LiDAR and LiDAR-Inertial Odometry Techniques	6
3.1.2 SLAM and Loop Closure-Based Localization	8
3.1.3 Localization Using Prior Maps	8
3.1.4 Point Cloud Registration Techniques	9
3.1.5 Dynamic Object Removal Techniques	10
3.2 Theoretical Background	11
3.2.1 Normal Distributions Transform (NDT)	11
3.2.2 Sensor Fusion Approaches	12
3.3 Research Gaps	15
4 Methodology	17

4.1	System Overview	17
4.1.1	LiDAR Scan Pre-Processing	19
4.1.2	Map Pre-Processing	20
4.2	LIDAR Inertia Odometry	21
4.3	Scan-to-Map Matching	22
4.4	Real-Time Factor Graph Optimization	23
4.4.1	Factor Graph Formulation	24
4.4.2	Maximum A Posteriori (MAP) Estimation	25
4.4.3	Fixed-Lag Smoothing and Marginalization	25
4.4.4	Outlier Rejection via Mahalanobis Distance	28
4.5	Dynamic Object Detection and Removal	28
4.5.1	Dynamic Object Detection	28
4.5.2	Detected Object removal	29
4.6	Environmental Noise Simulation	29
5	Experimental Setup and Result	31
5.1	Experimental Environment	31
5.1.1	Hardware and Software Setup	31
5.1.2	Datasets Collection and Map Preparation	32
5.2	Experimental Evaluation	33
5.2.1	Evaluation Metrics and Methodology	33
5.2.2	Localization Performance	34
5.3	Summary of Results	45
6	Discussion	46
6.1	Interpretation of Results	46
6.1.1	Accuracy and Robustness	46
6.1.2	Real-Time Performance	47
6.1.3	Environmental Adaptability	47
6.2	Implications	48
6.3	Limitations and Future Work	48
7	Conclusion	50
7.1	Summary of Findings	50
A	Trajectory Plot	51
	Bibliography	53

List of Figures

3.1	Architecture of FAST-LIO2: A tightly coupled LiDAR-Inertial Odometry [13]	7
3.2	Overview of CenterPoint 3d object detection framework. [27]	11
3.3	Factor graph representation: green nodes are state variables (poses), blue nodes are motion model factors, and red nodes represent sensor measurements.	14
4.1	System architecture of the proposed map-based localization pipeline. The system fuses high-frequency LiDAR-inertial odometry with global map-based pose corrections using scan-to-map matching and factor graph optimization.	18
4.2	Overview of LiDAR scan pre-processing steps. The raw point cloud undergoes spatial cropping, statistical outlier removal, voxel-grid downsampling, and dynamic object removal.	20
4.3	Demonstration of fixed-lag smoothing with a lag size of $n = 4$ in a factor graph-based localization system.	27
5.1	Hardware setup used for real-world data collection and testing: (a) Hunter robotic platform; (b) onboard sensors.	32
5.2	Saxion Sequence 01 – Trajectory Alignment with Zoomed Comparison	35
5.3	Absolute Pose Error Comparison of (a) Saxion Sequence 2 and (b) Saxion Sequence 3	36
5.4	Computation characteristics of the proposed localization system: (a) Sliding-window optimization scalability and (b) real-time frame-wise timing breakdown.	37
5.5	Top-down view of the localization trajectory overlaid on the pre-built 3D map. The red line indicates the estimated trajectory, while the white rectangle structures represent dynamically loaded local submaps within a fixed radius during runtime. This demonstrates how local map tiling supports scalable scan-to-map registration.	37
5.6	Detecting and removing dynamic objects from 3d LIDAR point-cloud	38
5.7	Point-cloud Registration Performance with and without Dynamic Object Removal	39
5.9	Error compression in transition zone	41

5.10	Point-cloud snapshots under increasing composite geometric noise levels: (a) Original scan; (b) Mild noise; (c) Moderate noise; (d) Severe noise.	42
5.11	Error comparison in transition zone	43
5.12	KITTI Sequence 05 – Trajectory Alignment with Zoomed Comparison	45
6.1	Comparison of sensor fusion architectures. (a) The current design integrates FAST-LIO2 with NDT-based map matching in a tightly coupled manner. (b) The proposed approach restructures the system into a modular factor graph, supporting IMU pre-integration and extensibility to visual or radar factors for increased robustness in degraded environments.	49
A.1	Saxion Sequence 02– Trajectory Alignment with Zoomed Comparison	51
A.2	Saxion Sequence 03 – Trajectory Alignment with Zoomed Comparison	52

List of Algorithms

1	LiDAR-Inertial Localization with Prior Map	19

List of Abbreviations

APE Absolute Pose Error

DoF Degrees of Freedom

EKF Extended Kalman Filter

FAST-LIO2 Fast LiDAR-Inertial Odometry 2

FOV Field of View

GNSS Global Navigation Satellite System

ICP Iterative Closest Point

IMU Inertial Measurement Unit

iSAM2 Incremental Smoothing and Mapping 2

LIDAR Light Detection and Ranging

LIO LiDAR-Inertial Odometry

MAP Maximum A Posteriori

MCL Monte Carlo Localization

NDT Normal Distributions Transform

NDT-OMP NDT with OpenMP (multithreaded) implementation

OMP Open Multi-Processing

PCL Point Cloud Library

RMSE Root Mean Square Error

ROS Robot Operating System

RTK Real-Time Kinematic

SLAM Simultaneous Localization and Mapping

STD Standard Deviation

TF Transform (coordinate transform in ROS)

Chapter 2

Introduction

2.1 Motivation

High-precision and reliable localization is crucial for autonomous vehicles and robots, as it directly affects tasks like path planning, obstacle avoidance, and other repetitive missions [1]. Typically, positioning relies on Global Navigation Satellite Systems (GNSS) and Inertial Navigation Systems (INS), which complement each other. GNSS can provide absolute position without drift especially when using differential signals while INS delivers frequent updates to track movement[2]. However, in urban areas with tall buildings or indoor environments, GNSS signals can be blocked or distorted, and INS alone may accumulate significant drift over time[3]. Moreover, these systems do not inherently provide information about the surrounding environment.

To address such limitations, researchers have adopted onboard sensors like LiDAR, monocular cameras, and stereo cameras. One widely used method is Simultaneous Localization and Mapping (SLAM), which enables a robot to estimate its position in a completely new environment without needing a prior map. While SLAM is flexible and powerful, it can drift if loop closures are infrequent or the robot cannot revisit known areas to correct its estimates.

An alternative that improves accuracy and stability is map-based localization, where a pre-built map (created beforehand by SLAM, GNSS-based surveys, or other methods) is used as a global reference [4]. In this approach, the robots's current sensor data such as LiDAR scans are matched against the detailed information in the map to determine its position. This matching process helps minimize drift and provides robust localization, even in environments where GNSS signals are unavailable or unreliable. Once the map is constructed, indoor and outdoor autonomous systems can repeatedly use it to perform their tasks more reliably, whether that involves delivering goods, patrolling a facility, or

navigating urban streets. As a result, map-based localization is considered an indispensable component of future autonomous navigation, especially when consistent high accuracy and global reference information are required.

Despite its advantages, map-based localization poses several challenges. First, real-time performance requires efficient management of high-resolution 3D maps and the use of fast, precise scan-to-map matching algorithm for accurate pose estimation. Second, integrating local motion estimation (e.g., via LiDAR-Inertial Odometry) with global map alignment must be handled carefully to ensure both consistency and computational feasibility. Finally, the presence of dynamic objects and adverse environmental conditions further complicates the localization pipeline.

This thesis addresses these challenges by proposing a hybrid system that combines real-time LiDAR-Inertial Odometry (FAST-LIO2), efficient map-matching via multithreaded NDT (NDT-OMP), and a sliding-window factor graph fusion approach. The system dynamically loads submaps around the robot based on a configurable radius, allowing scalable scan-matching while maintaining global consistency. In addition, a learning-based object detection module identifies and removes dynamic objects from the LiDAR scans, further enhancing robustness. Together, these components form a real-time, accurate, and robust localization framework tailored for large-scale, semi-dynamic environments.

2.2 Objective

The primary objective of this thesis is to develop a real-time, LiDAR-based localization framework that achieves both high-frequency motion estimation and drift-free global consistency in large-scale environments. This is accomplished by integrating LiDAR-Inertial Odometry with map-based scan matching, and fusing them through a computationally efficient sliding-window factor graph.

- To integrate a robust LiDAR-Inertial Odometry(LIO) module using FAST-LIO2, capable of providing real-time pose estimates by tightly coupling LiDAR and IMU data.
- To develop a scan-to-map matching module by integrating a multithreaded Normal Distributions Transform (NDT-OMP) algorithm, enabling real-time alignment of incoming LiDAR scans with a pre-built 3D point cloud map.
- To fuse LiDAR-Inertial Odometry(LIO) and map matching results using a sliding-window factor graph optimization approach, enabling efficient, bounded optimization that retains local consistency while discarding outdated states.

- To integrate a learning-based dynamic object detection module into the LiDAR processing pipeline.
- To design and implement a grid-based submap management system that loads only the relevant local tiles within a configurable radius of the robot's position and incrementally replaces out-of-range tiles without reloading the full map.
- To evaluate the proposed system on benchmark datasets and custom scenarios, measuring performance in terms of localization accuracy, computation time, memory usage, and robustness under varying environmental conditions.

2.3 Research Questions

The primary questions guiding this research are:

1. How can LiDAR-Inertial odometry (FAST-LIO2) and NDT-based map matching be effectively fused to enhance real-time localization accuracy and robustness?
2. How does integrating a prior map improve the accuracy and robustness of LiDAR-based localization in GNSS-challenged environments?
3. How does the proposed system maintain real-time performance and scalability in large-scale environments?
4. Does the proposed localization method maintain accurate localization in feature sparse and noisy environmental condition?

2.4 Scope and limitations

2.4.1 Scope

The proposed localization system is designed for real-time operation in diverse environments, including indoor spaces, outdoor areas, and urban environments. It assumes the availability of a high-resolution 3D LIDAR map of the operation environment. Furthermore, the system is capable of maintaining localization even when the robot temporarily traverses boundary or transition zones where the pre-built map is sparse or unavailable. The system provides accurate pose estimation by fusing LIDAR-Inertial Odometry with scan-to-map matching, and its particularly suited scenarios where GNSS is unreliable or unavailable.

2.4.2 Limitations

- The system does not perform global localization (i.e., it cannot determine an unknown initial pose within the map). It assumes the robot starts with a known or approximated initial position close to its true location.

1.4 Contributions of the Thesis

This thesis contributes to the advancement of real-time, map-based localization in autonomous systems by addressing key challenges related to accuracy, robustness, and scalability in GNSS-denied and semi-dynamic environments. The main contributions of this work are summarized as follows:

1. **Hybrid Localization Framework:** Developed a real-time localization pipeline that combines FAST-LIO2 odometry with NDT-based scan-to-map matching within a sliding-window factor graph, achieving drift-resilient 6-DoF pose estimation.
2. **Dynamic Submap Loading:** Implemented an adaptive, tile-based map management strategy that loads only relevant submaps based on the robot's estimated position, ensuring scalability and memory efficiency in large-scale environments.
3. **Learning-Based Dynamic Object Removal:** Integrated a deep-learning-based CenterPoint detector into the LiDAR preprocessing stage to identify and remove dynamic objects, improving the robustness of scan matching in semi-dynamic scenes.
4. **Real-Time Performance:** Achieved an average per-frame processing latency below 23 milliseconds through multithreaded NDT registration, voxel filtering, and efficient scan-to-map alignment.
5. **Robustness in Degraded Conditions:** Demonstrated reliable localization performance in feature-sparse areas and under moderate visibility degradation (e.g., fog), with fallback to high-rate odometry when scan matching becomes unreliable.

2.5 Overview of the thesis structure

This thesis is organized as follows. **Chapter 2** reviews relevant literature on LiDAR-based localization, covering LiDAR-Inertial Odometry, SLAM, map-based methods, scan matching, sensor fusion, and learning-based dynamic object removal. **Chapter 3** details the proposed system architecture, outlining the design and integration of each module.

Chapter 4 presents the experimental setup and results, including performance metrics and comparisons with baseline methods. **Chapter 5** discusses key findings, addresses system limitations, and outlines potential directions for improvement and future work. Finally, **Chapter 6** concludes the thesis by summarizing the main contributions.

Chapter 3

Litreature Review

3.1 Overview of Relevant Literature

This section provides an overview of the foundational literature relevant to the design and development of the proposed localization system. It reviews key concepts and techniques in LiDAR and LiDAR-Inertial Odometry, SLAM, map-based localization, point cloud registration, and dynamic object detection using 3D LiDAR data.

3.1.1 LiDAR and LiDAR-Inertial Odometry Techniques

Localization is a crucial capability for mobile robots, referring to the process of determining the robot's position and orientation in a given environment. Early attempts at localization depended heavily on wheel encoders, commonly known as wheel odometry, but this method suffered from significant drift whenever the wheels slipped or the ground was uneven [5]. As robotics advanced, researchers explored range sensors and visual sensors to improve motion estimation [6]. At the same time, algorithms like Iterative Closest Point (ICP) emerged, enabling the alignment of consecutive scans and sparking a separate stream of research in range sensor-based odometry [7].

Building on these ideas, LiDAR-only odometry gained popularity due to LiDAR's immunity to lighting changes and ability to capture dense 3D data. One key approach uses straightforward scan matching via ICP [7], but such methods can struggle in environments with few unique geometric features. To address this, feature-based systems extract distinctive cues from each LiDAR scan, as demonstrated in LOAM [8], where sharp edges and planar surfaces guide the matching process. While LiDAR-only odometry can produce accurate short-term estimates, it may drift over time if the robot moves rapidly or the environment is mostly repetitive.

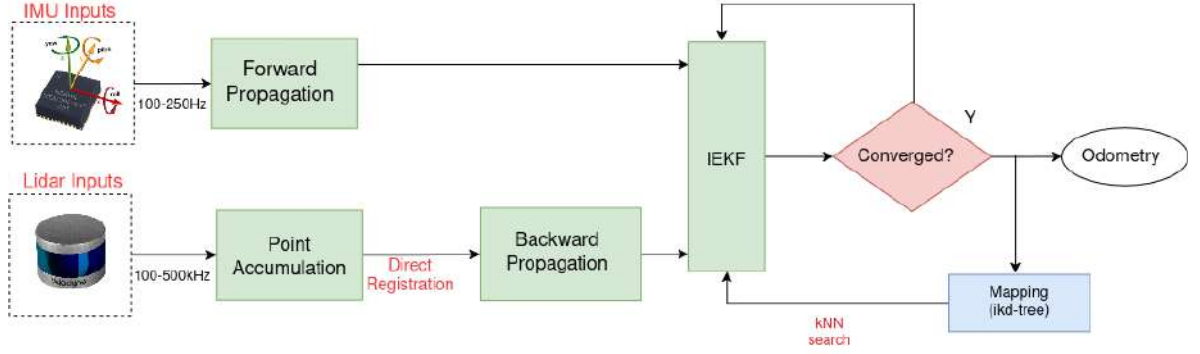


Figure 3.1: Architecture of FAST-LIO2: A tightly coupled LiDAR-Inertial Odometry [13]

To reduce long-term drift, many researchers incorporate IMU (Inertial Measurement Unit) data, resulting in what is called LiDAR-Inertial Odometry (LIO). In a loosely coupled setup, LiDAR and IMU each compute their own pose, and then a higher-level filter (like an Extended Kalman Filter) merges these estimates [9]. This approach is easier to design because the LiDAR odometry module and the IMU integration module work somewhat independently. However, the separate modules might not fully exploit each other's measurements.

By contrast, tightly coupled designs feed raw IMU data directly into the LiDAR optimization. For instance, LIO-SAM employs a factor graph where both LiDAR and IMU preintegration factors are updated in the same framework, allowing more constraints to be used in each calculation [10]. As graph-based LiDAR-inertial odometry solutions grew more complex and demanding, researchers began exploring filter-based methods built around the Kalman filter. One notable example is LINS [11], which uses an iterated Error State Kalman filter (iESKF) to speed up pose estimation; however, it still faces performance bottlenecks when dealing with large numbers of LiDAR points, since computing the Kalman gain for every point can be costly. In response, FAST-LIO [12] introduced a more efficient Kalman gain calculation that significantly lowers this overhead. Later, As shown in Figure 3.1 FAST-LIO2 [13] expanded on that approach by completely removing the feature extraction step and matching raw LiDAR points directly to the map. To accelerate nearest-neighbor queries, it employs a specialized data structure called an ikd-Tree, allowing the system to achieve both high accuracy and even faster run.

Due to its high computational efficiency, accuracy, and real-time capability, FAST-LIO2 is adopted in this thesis as the LiDAR-Inertial Odometry (LIO) front-end. Its design aligns with the system's objective to maintain high-frequency odometry while minimizing drift, providing a solid foundation for downstream scan-to-map correction and graph-based fusion.

3.1.2 SLAM and Loop Closure-Based Localization

Moving beyond pure odometry, SLAM (Simultaneous Localization and Mapping) takes an extra step by adding loop closure mechanisms to maintain global consistency [14]. While LiDAR-Inertial Odometry can reduce local drift, it remains prone to cumulative error over long distances if no global corrections are made. By detecting revisits to previously mapped areas, a full SLAM pipeline like Google Cartographer (for 2D LiDAR) [15] or other graph-based back ends [16] refines the entire trajectory, resulting in notably higher accuracy. Recent SLAM systems like MOLA slam [17] and KISS-ICP[18] have further advanced LiDAR-based localization by improving loop closure detection and trajectory optimization for large-scale environments. However, these systems carry greater computational cost and design complexity, since maintaining a global graph or submap structure calls for advanced data management.

SLAM inherently relies on loop-closure events or prior constraints to limit its global drift; if these loop closures are sparse or absent over a long trajectory, the system can still accumulate significant positioning errors [16, 14]. That limitation underscores the motivation for map-based localization, in which an existing map is used to provide absolute constraints and reduce drift, even in lengthy or repetitive environments where loop closures are not guaranteed.

3.1.3 Localization Using Prior Maps

Estimation of the position of a sensor on a map is essential for navigation systems. While several kinds of map representations are used, depending on the use scenario 3D point cloud maps are among the most popular representations owing to their simplicity and expressiveness.[19]Because constructing a point cloud map is relatively easy with recent precise range sensors and mapping algorithms, point cloud maps are used for a wide range of applications, from indoor service robots to outdoor driving of autonomous vehicles.

Monte Carlo Localization (MCL) is a well-established probabilistic technique for 2D map-based localization [20]. It excels in handling non-linear and non-Gaussian systems, offering robustness to ambiguous observations and multi-modal distributions. However, its application in 3D localization is rare due to the exponential increase in the number of particles required to represent the full 6-DoF state space. This makes real-time use in point cloud-based 3D environments computationally prohibitive

Multiple recent works highlight the potential of map-aided localization for achieving robust and precise vehicle pose estimation.[21] propose building a 2D road-surface reflectivity map from lidar data, then using real-time lidar scan correlation against this map to localize

within a few decimeters in busy urban areas. Similarly, [3] adopt a detailed segmentation approach: each incoming lidar frame is broken down into ground, curb, edge, and surface features, which are then aligned with a prior feature map; the refined estimates are further fused with a MEMS-IMU for centimeter-level accuracy at high frequency.

By contrast, [22] utilize stereo-based “visual point cloud” maps, avoiding LiDAR hardware yet involving dense stereo matching for both the offline (map) and online (frame) steps. Their system then registers these point clouds via normal distribution transform (NDT). While it lowers sensor costs, it adds a heavy stereo workload and remains susceptible to lighting or texture issues. The authors also report that deviations can occur at sharp turns, albeit these are later corrected. Finally, [23] integrate LOAM odometry with a global segment-based prior map (via SegMap) for re-localization when drift accumulates. The system occasionally triggers place recognition in the global map, confirms geometry with RANSAC, then refines the pose using fine-grained ICP.

By contrast, [22] utilize stereo-based “visual point cloud” maps, avoiding LiDAR hardware yet involving dense stereo matching for both the offline (map) and online (frame) steps. Their system then registers these point clouds via normal distribution transform (NDT). While it lowers sensor costs, it adds a heavy stereo workload and remains susceptible to lighting or texture issues. The authors also report that deviations can occur at sharp turns, albeit these are later corrected. Similarly, [1] propose a stereo vision localization method within 3D LiDAR maps by extracting visual features in the Bird’s-Eye View (BEV) domain and matching them to the LiDAR map. Finally, [23] integrate LOAM odometry with a global segment-based prior map (via SegMap) for re-localization when drift accumulates. The system occasionally triggers place recognition in the global map, confirms geometry with RANSAC, then refines the pose using fine-grained ICP.

Together, these works illustrate the diversity of map-based localization strategies, but many are constrained by computational inefficiency, dependence on handcrafted or road-specific map features, and limited scalability or adaptability to dynamic and degraded environments. This underscores the need for a more efficient, generalizable, and real-time-capable localization framework.

3.1.4 Point Cloud Registration Techniques

Point cloud registration is a key technique in LiDAR-based localization, used to align a current LiDAR scan with a prior map to estimate the sensor’s pose. The classic Iterative Closest Point (ICP) algorithm performs this by minimizing distances between matched point pairs across the two clouds. While ICP is conceptually simple, it suffers from sensitivity to

noise, poor initial alignment, and local minima, making it unreliable for large-scale or noisy environments [7].

To address these limitations, the Normal Distributions Transform (NDT) was proposed by [24] and further elaborated in subsequent research[25]. NDT offers a compact representation of surfaces by converting point clouds into a set of local probability density functions (PDFs), each characterizing a surface section. Instead of relying on discrete point matches, NDT represents the map as a grid of voxels, each containing a Gaussian distribution that models the local geometry. Scan points are registered by maximizing their likelihood under these distributions. This makes NDT more robust to outliers, partial overlaps, and noisy measurements.

Modern implementations such as NDT-OMP further improve runtime performance through multi-threading, enabling real-time scan-to-map matching in large-scale environments [26]. Due to its robustness and efficiency, NDT is widely used in autonomous driving and is adopted in this thesis for scan-matching against a pre-built point cloud map.

3.1.5 Dynamic Object Removal Techniques

In LiDAR-based localization, scan-matching techniques are widely employed to align real-time sensor data with a pre-built static map. However, dynamic objects such as moving vehicles, pedestrians, and cyclists introduce discrepancies between the current point cloud and the static reference. These inconsistencies can significantly degrade localization performance, leading to pose drift, poor convergence, or failure in highly dynamic environments.

To enhance the reliability of scan-matching, dynamic object removal has been introduced as a crucial preprocessing step. Early methods, including Euclidean clustering, geometric filtering, and background subtraction, identified dynamic elements based on heuristics such as size, shape, or temporal consistency [3, 26]. Although computationally efficient, these approaches often struggle in cluttered or semi-structured environments, resulting in high false positive rates and poor generalization. Recent advancements leverage deep learning-based 3D object detection models such as PointPillars[28], SECOND [29], and CenterPoint [27] to semantically identify dynamic agents. By predicting 3D bounding boxes for known classes, these models enable precise removal of dynamic objects, preserving static features critical for accurate localization.

Among these, CenterPoint[27] has emerged as a state-of-the-art solution. It employs a center-based detection paradigm over voxelized LiDAR input and achieves high accuracy on large-scale datasets such as nuScenes[30], making it particularly suitable for real-time dynamic object filtering.

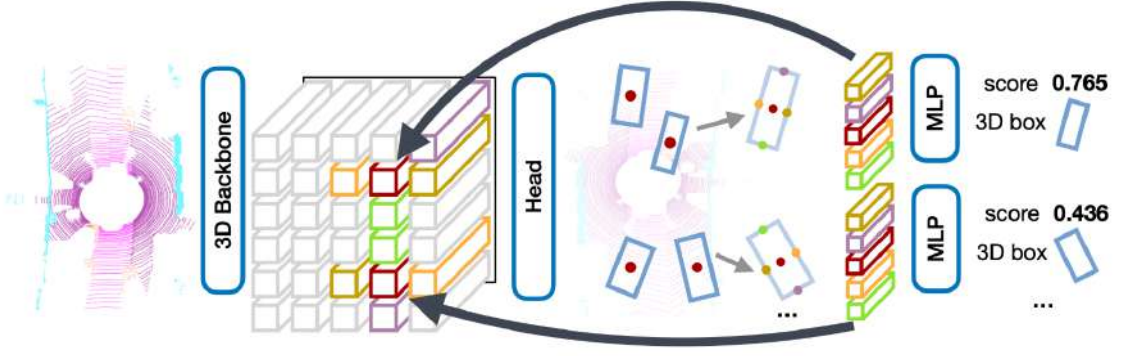


Figure 3.2: Overview of CenterPoint 3d object detection framework. [27]

CenterPoint follows a two-stage architecture[27]. First, the raw LiDAR point cloud is voxelized into a sparse 3D tensor, allowing efficient processing by sparse convolutional neural networks. A backbone network typically based on PointPillars [28] or SECOND[29] extracts spatial features while maintaining the geometric structure of the environment. These features are projected onto a Bird’s-Eye View (BEV) representation, preserving spatial relationships and enabling faster 2D convolutional operations. In the second stage, a keypoint-based detection head predicts object centers by generating a center heatmap over the BEV feature map. Following center localization, the model regresses the full set of object attributes, including 3D bounding box dimensions, orientation (yaw angle), velocity, and semantic class label. This decoupled center-to-attribute approach simplifies the detection task and enhances robustness against small, distant, or partially occluded objects, as illustrated in Figure 3.2. Due to its high detection accuracy and real-time inference capability, CenterPoint is adopted in this work as the dynamic object detection module.

3.2 Theoretical Background

3.2.1 Normal Distributions Transform (NDT)

The NDT algorithm begins by subdividing the space occupied by a scan into discrete grid cells—squares in 2D or cubes in 3D. A PDF is calculated for each cell based on the distribution of points within that cell. Each PDF describes how points within the cell are likely generated, assuming the points result from a multivariate normal random process. The probability density function for a cell is defined as:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right) \quad (3.1)$$

where μ and Σ denote the mean vector and covariance matrix of the reference of the

scan surface points within the cell where $\mathbf{x}$ lies. The mean and the covariance are computed as

$$\mu = \frac{1}{m} \sum_{k=1}^m \mathbf{y}_k \quad (3.2)$$

$$\Sigma = \frac{1}{m-1} \sum_{k=1}^m (\mathbf{y}_k - \mu)(\mathbf{y}_k - \mu)^T \quad (3.3)$$

where $y_k = 1, \dots, m \dots$ are the positions of the reference scan points contained in the cell.

This probabilistic representation gives a piecewise smooth surface approximation with continuous derivatives, capturing the local surface position, orientation, and smoothness effectively.

When applying NDT for scan registration, the objective is to determine the pose of the current scan that maximizes the likelihood that scan points align with the reference map surface. The parameters to be optimised; that is, the rotation and translation of the pose estimate of the current scan, can be encoded in a vector $\vec{p}$. The current scan is represented as a point cloud $X = \{\vec{x}_1, \dots, \vec{x}_n\}$. Assume that there is a spatial transformation function $T(\vec{p}, \vec{x})$ that moves a point $\vec{x}$ in space by the pose $\vec{p}$. Given some PDF $\vec{p}$ for scan points, the optimal pose maximizes the likelihood function:

$$\Psi = \prod_{k=1}^n p(T(\mathbf{p}, \mathbf{x}_k)) \quad (3.4)$$

or equivalently minimizes the negative log-likelihood of Ψ

$$-\log \Psi = - \sum_{k=1}^n \log p(T(\mathbf{p}, \mathbf{x}_k)) \quad (3.5)$$

The primary distinction between 2D and 3D NDT algorithms resides in the spatial transformation function and its partial derivatives. In 2D, rotations are represented by a single angle, while in 3D, rotations require more complex representations (rotation matrices or quaternion representations).

3.2.2 Sensor Fusion Approaches

Filtered Based Approaches

Filtering-based sensor fusion recursively estimates a system's state using two main phases: a prediction step (where the state is propagated using a motion model) and an update step (where incoming sensor measurements refine that prediction) [20]. This real-time,

sequential estimation framework is widely employed in robotics to handle noisy sensors while maintaining tractable computational load. Nonetheless, large global corrections or loop closures can be challenging to incorporate without more advanced smoothing or factor-graph techniques[14].

1. **Extended Kalman Filter(EKF)** extends the linear Kalman Filter [31] to handle nonlinear systems by linearizing the motion and measurement models around the current estimate. It is widely used due to its computational simplicity, although linearization can degrade accuracy in highly nonlinear settings [20].
2. **Error State Kalman Filter(ESKF)** focuses on estimating the deviation between a nominal trajectory and the true state [32]. A separate routine generates a continuous nominal estimate, while the ESKF maintains small error states for position, orientation, and sensor biases:

$$\delta \mathbf{x}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k^{\text{nominal}}. \quad (3.6)$$

By filtering only $\delta \mathbf{x}_k$, it improves numerical stability in rotation and bias terms, and is particularly effective for high-rate IMU fusion (e.g., in FAST-LIO [12] or LINS [11]). Major loop closures or global corrections remain difficult in a strictly forward-update framework [14].

3. **Unscented Kalman Filter(UKF)** avoids explicit Jacobian linearization by employing a deterministic sampling approach (sigma points) [33]. A set of points $\{\chi_i\}$ is chosen around the mean and propagated through the nonlinear functions $\mathbf{f}$ and $\mathbf{h}$. The resulting transformed set describes the posterior distribution more accurately than a simple first-order expansion. While the UKF handles significant nonlinearities better than the EKF, it can be more computationally demanding [20].

Factor Graph Based Sensor Fusion

Factor graphs are a powerful framework for multi-sensor fusion in robotics, enabling the simultaneous estimation of a robot's trajectory over time. Unlike filtering approaches which operate recursively, factor graphs formulate state estimation as a global optimization problem, making them well-suited for handling nonlinearity, drift correction, and loop closures efficiently [14, 34].

A factor graph is an undirected bipartite graph that separates variable nodes (robot poses) from factor nodes (motion constraints or sensor measurements) [35]. As shown in Figure 3.3, green nodes represent state variables, while red and blue nodes represent measurement and motion constraints, respectively. Each factor connects only to the variables it

influences, forming a sparse, structured representation that is computationally efficient for optimization.

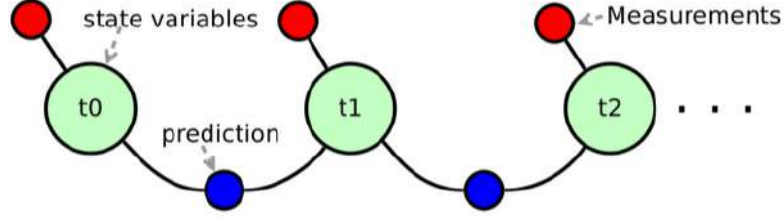


Figure 3.3: Factor graph representation: green nodes are state variables (poses), blue nodes are motion model factors, and red nodes represent sensor measurements.

Mathematically, a factor graph models the joint posterior distribution over all states X , given observations Z , as a product of smaller, local functions:

$$P(X | Z) \propto \prod_k \phi_k(X_{\alpha(k)}), \quad (3.7)$$

where ϕ_k is a factor derived from a sensor model or prior, and $X_{\alpha(k)}$ is the subset of state variables relevant to that factor.

In the context of localization, let $X = \{x_0, x_1, \dots, x_n\}$ be the robot's pose trajectory and $Z = \{z_1, \dots, z_M\}$ be the set of sensor measurements. The joint posterior can be factorized as:

$$P(X | Z) \propto P(x_0) \prod_{k=1}^n P(x_k | x_{k-1}) \prod_{m=1}^M P(z_m | X_{\alpha(m)}), \quad (3.8)$$

where:

- $P(x_0)$ is the prior on the initial state,
- $P(x_k | x_{k-1})$ is the motion model (e.g., from IMU preintegration),
- $P(z_m | X_{\alpha(m)})$ is the sensor measurement model (e.g., LiDAR scan matching).

Assuming Gaussian noise, the Maximum A Posteriori (MAP) estimation reduces to a nonlinear least-squares problem:

$$X^* = \arg \min_X \sum_k \|h_k(X_{\alpha(k)}) - z_k\|_{\Sigma_k}^2, \quad (3.9)$$

where $h_k(\cdot)$ is the measurement function, z_k is the observation, and Σ_k is the associated noise covariance matrix.

Incremental Solvers

For real-time or large-scale applications, solving the entire problem from scratch at each step may be intractable [20]. Incremental smoothing algorithms like iSAM2 [36] maintain a factorization of the problem in a data structure (often a Bayes tree) that can be efficiently updated with new measurements. Rather than re-solving globally each time, iSAM2 carries out local re-linearization and partial variable reordering, providing near-constant-time updates for many practical cases. Furthermore for Bounded computational cost, a sliding-window strategy are used to optimize only a fixed set of recent states by discarding older variables while preserving accuracy [37]

3.3 Research Gaps

The literature review highlights several gaps and challenges in existing LiDAR-based localization systems, particularly in the context of real-time performance, scalability, and robustness. These gaps inform the design and objectives of the proposed system. Key identified gaps include:

- **Long-Term Drift and Global Consistency:** Existing LIO and SLAM often suffer from long-term drift and loss of global consistency, particularly in GNSS-denied environments. While SLAM approaches attempt to address this via loop closures, their effectiveness is limited in environments with sparse or delayed loop detection.
- **Computational Intractability of MCL:** Monte Carlo Localization (MCL) is robust for 2D localization but computationally impractical for 3D due to the exponential increase in required particles. This limits its application in dense 3D point cloud and 6-DoF pose estimation.
- **Real-Time Performance and Scalability:** map-based localization approaches struggle with real-time performance and scalability in large maps. Standard algorithms like ICP and NDT often fail to maintain real-time performance as map size increases, leading to delayed convergence or misalignment.
- **Generalization:** Existing map-based approaches rely on handcrafted features, reflectivity profiles, or selected geometric structures from street-level maps limiting their generalization across diverse environments. These systems often fail to generalize across diverse environments and degrade in areas with low geometric richness, unmapped zones, or transitional boundary regions.

- **Dynamic Object Handling:** Existing systems assume static environments and do not systematically address the impact of dynamic objects on scan matching. Dynamic object removal is rarely incorporated in real-time.
- **Environmental Degradation:** Adverse conditions such as fog or sensor noise are often underrepresented in evaluation protocols, leaving a gap in understanding system resilience under realistic field conditions.

Chapter 4

Methodology

This chapter outlines the methodological foundation of the proposed localization system. It describes the main system components and how they are integrated to estimate the robot's pose by combining LiDAR and inertial measurements with a pre-built 3D map. The focus is on the overall process and key implementation choices that support accurate and real-time localization.

4.1 System Overview

The proposed localization system estimates the 6-DoF pose of a mobile robot by combining LiDAR and inertial sensor data with a prior 3D map of the environment. The system is designed to operate in real time and deliver accurate, drift-resilient localization in complex, large-scale, and partially dynamic environments.

The localization pipeline begins with preprocessing of the raw LiDAR point cloud. This includes filtering, downsampling, and transformation to a consistent reference frame. To improve robustness in semi-dynamic scenes, transient objects such as vehicles and pedestrians are detected using a deep learning-based 3D object detection model and removed from the scan prior to registration.

In parallel, inertial measurements from the IMU are fused with deskewed LiDAR data using the FAST-LIO2 framework [13], which provides high-frequency odometry estimates through tightly coupled LiDAR-inertial integration. This module ensures locally consistent motion tracking with minimal latency.

To correct drift and maintain alignment with a global reference, the system performs scan-to-map matching. A local submap is dynamically loaded from the global map based on the robot's current estimated position. Only relevant tiles within a specified radius are retrieved, optimizing memory and computation. The filtered LiDAR scan is then aligned

with this submap using a multithreaded Normal Distributions Transform (NDT-OMP) algorithm [26], which produces a globally referenced pose estimate.

Both odometry and scan-matching results are fused using a fixed-lag smoothing approach based on factor graph optimization. This module maintains a sliding window of recent trajectory states and performs real-time optimization to output a drift-resilient and globally consistent pose estimate.

The overall system architecture and step-by-step algorithm flow are illustrated in Figure 4.1 and Algorithm 1, respectively. The subsequent sections describe each component of the system in detail, including LiDAR preprocessing, local submap loading, odometry estimation, scan matching, graph-based fusion, and dynamic object removal.

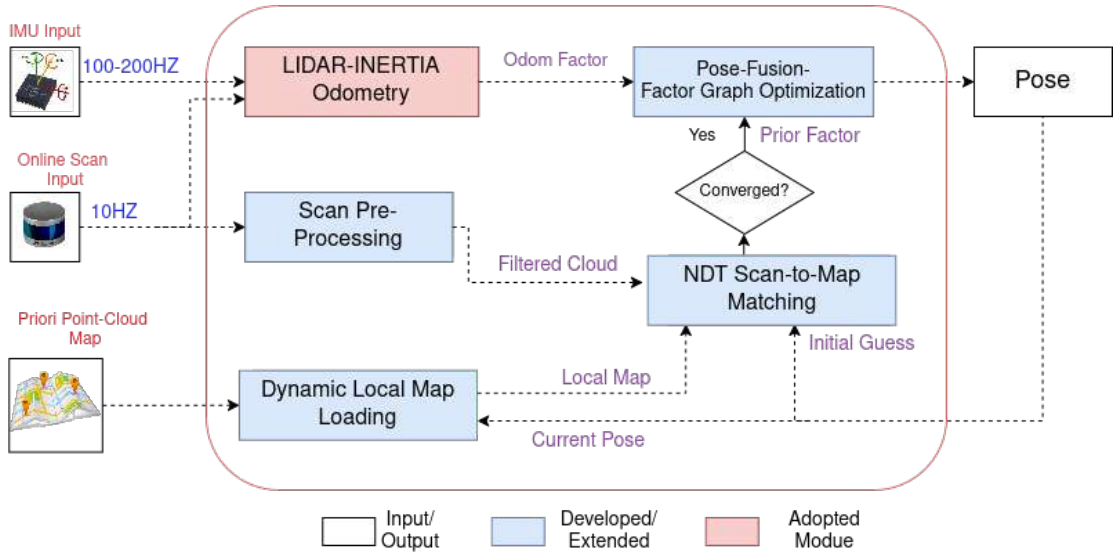


Figure 4.1: System architecture of the proposed map-based localization pipeline. The system fuses high-frequency LiDAR-inertial odometry with global map-based pose corrections using scan-to-map matching and factor graph optimization.

Algorithm 1 LiDAR-Inertial Localization with Prior Map**Input:** $\mathcal{P}_t$ (LiDAR Point Cloud), $\mathcal{I}_t$ (IMU Data), $\mathcal{M}$ (Prior Map)**Output:** $\mathbf{T}_t$ (Real-time 6-DoF Pose)

```

1: while System is running do
2:   Acquire LiDAR point cloud  $\mathcal{P}_t$  and IMU data  $\mathcal{I}_t$ 
3:   Perform scan pre-processing on  $\mathcal{P}_t$  to obtain filtered point cloud  $\mathcal{P}_t^{filtered}$ 
4:   Estimate LiDAR-Inertial Odometry using  $\{\mathcal{P}_t^{filtered}, \mathcal{I}_t\}$  to obtain odometry pose  $\mathbf{T}_t^{lio}$ 
5:   if  $\|\mathbf{T}_t - \mathbf{T}_{last}^{map}\| > d_{threshold}$  then
6:     Load dynamic local map  $\mathcal{M}_t^{local}$  from  $\mathcal{M}$  centered at  $\mathbf{T}_t$ 
7:     Update  $\mathbf{T}_{last}^{map} \leftarrow \mathbf{T}_t$ 
8:   end if
9:   Perform scan-to-map matching between  $\mathcal{P}_t^{filtered}$  and  $\mathcal{M}_t^{local}$  to obtain map-based pose  $\mathbf{T}_t^{map}$ 
10:  if Matching score  $> s_{threshold}$  then
11:    Add prior factor with  $\mathbf{T}_t^{map}$  to factor graph
12:  end if
13:  Add odometry factor with  $\mathbf{T}_t^{lio}$  to factor graph
14:  Optimize sliding window factor graph to obtain final pose  $\mathbf{T}_t$ 
15: end while

```

4.1.1 LiDAR Scan Pre-Processing

Raw LiDAR scans typically produce large point-clouds that vary in density across the range of the sensor. While points near the sensor are densely sampled, those at longer distances become increasingly sparse due to fixed angular resolution. This non-uniform density, combined with measurement noise and dynamic objects, introduces challenges for efficient and robust localization. To address these issues, I apply a structured pre-processing pipeline that filters, reduces, and cleans the incoming data prior to further processing.

As illustrated in Figure 4.2, the following steps are applied to each incoming point cloud:

- **Crop-Box Filtering:** I use a spatial crop-box filter to remove points outside a user-defined region of interest centered around the sensor. This step eliminates distant or irrelevant points, reducing computational load and focusing on the region critical for accurate scan alignment.
- **Statistical Outlier Removal:** To mitigate the effect of random noise, I apply a statistical filter that removes isolated or spurious measurements. For each point, the average distance to its k nearest neighbors is computed. Points exceeding the global mean by a threshold multiple of the standard deviation are discarded as outliers.
- **Voxel-Grid Downsampling:** To reduce data size while preserving geometric structure, I downsample the filtered point cloud using a voxel-grid filter. The space is

partitioned into uniform voxels, and all points within each voxel are replaced by their centroid. This ensures that the point cloud remains computationally tractable without compromising registration fidelity.

- **Dynamic Object Removal:** To improve robustness in dynamic environments, points associated with moving objects are removed. I use a learning-based detector (CenterPoint) to identify dynamic agents and remove their corresponding points using 3D bounding boxes. Further details are provided in Section 4.5.

This cleaned and downsampled point-cloud is then used as input to the scan-to-map matching module, where it is aligned with a local submap to estimate the robot’s global pose.

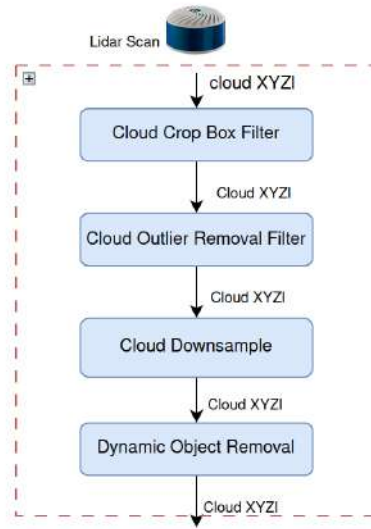


Figure 4.2: Overview of LiDAR scan pre-processing steps. The raw point cloud undergoes spatial cropping, statistical outlier removal, voxel-grid downsampling, and dynamic object removal.

4.1.2 Map Pre-Processing

To support real-time localization in large-scale environments, the global point-cloud map is pre-processed into a 2D grid of tiles each stored as an individual point-cloud file. During this offline division, every point’s x - y coordinates are used to assign it to a grid cell of side length s , where s is chosen to balance tile granularity against file count. Each grid cell is saved as an individual point cloud file, and its index is stored as metadata for fast retrieval. During localization, these indexes enable loading only the relevant grids within the robot’s local map radius, reducing memory usage and improving processing efficiency.

At runtime, I maintain a dynamic local sub-map around the robot by loading only those tiles whose centers lie within a radius

$$r_{\text{map}} = r_{\text{lidar}} + r_{\text{margin}}, \quad (4.1)$$

where r_{lidar} is the LiDAR's maximum range and r_{margin} is a small buffer to ensure sufficient overlap for robust scan-to-map matching. To avoid unnecessary I/O, I trigger a map update only when the robot's current position $\mathbf{T}_t$ moves farther than a threshold d_{thresh} from the last update location $\mathbf{T}_{\text{last}}$, that is,

$$\|\mathbf{T}_t - \mathbf{T}_{\text{last}}\|_2 > d_{\text{thresh}}. \quad (4.2)$$

When equation 4.2 is satisfied, the system identifies all grid cells whose centers fall within the circle of radius r_{map} about $\mathbf{T}_t$. Those tiles not yet in memory are loaded from the global map, while any previously loaded tiles that now lie outside this radius are evicted. Finally, $\mathbf{T}_{\text{last}}$ is updated to the new robot position. This selective, radius-based loading and unloading scheme keeps the local map size bounded, minimizes memory usage, and ensures that scan-to-map matching always operates on the most relevant portion of the prior map.

4.2 LIDAR Inertia Odometry

FAST-LIO2 serves as the high-frequency LiDAR-Inertial Odometry front-end in our localization pipeline. At each timestep t , the filter ingests IMU angular rate $\omega_m(t)$ and linear acceleration $a_m(t)$ (200 Hz) together with motion-compensated LiDAR point clouds (10 Hz), deskewed using the same IMU measurements. All sensor data are transformed into a common body frame via the known extrinsic calibration

$$T_{\text{imu} \rightarrow \text{lidar}} \in SE(3). \quad (4.3)$$

FAST-LIO2 implements a tightly-coupled iterated Kalman filter, directly associating each incoming LiDAR point with the local map using an incremental k-d tree (ikd-Tree), thereby eliminating explicit feature extraction and frame-to-frame scan matching. At each LiDAR update, the filter propagates the state estimate $\mathbf{x}(t)$ via the IMU pre-integration equations, then corrects it using the residuals between measured points and their expected positions in the map. The result is a 6-DOF pose estimate

$$\hat{\mathbf{T}}_{WB}(t) = \begin{bmatrix} \mathbf{R}_{WB}(t) & \mathbf{t}_{WB}(t) \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3), \quad (4.4)$$

which we publish as an odometry factor in our global factor graph. IMU noise densities and bias random-walk coefficients $(\sigma_\omega, \sigma_a, \sigma_{b_\omega}, \sigma_{b_a})$ are set according to the manufacturer’s specifications. While FAST-LIO2 exhibits negligible drift over short intervals, residual error on long trajectories is corrected through map matching constraints in the subsequent optimization (Section 4.4).

4.3 Scan-to-Map Matching

The Scan-to-Map Matching module is a critical component of the proposed localization system. Its primary objective is to align the current LiDAR scan with a dynamically loaded local submap and estimate the robot’s pose relative to the prior map coordinate frame. Several registration techniques were considered during system design. While Iterative Closest Point (ICP)[7] and the standard Normal Distributions Transform (NDT)[24] are commonly used, they were not selected due to their high computational cost and sensitivity to initial pose estimates.

To overcome these limitations, this work adopts a multithreaded implementation of the Normal Distributions Transform (NDT-OMP) [26], which leverages OpenMP-based parallelization to significantly accelerate computation. NDT-OMP maintains the robustness of NDT while enabling real-time performance through parallel evaluation of gradients and Hessians during scan registration. This makes it particularly well-suited for large-scale and time-constrained localization tasks.

The module begins by voxelizing the local map: each occupied cell i is represented by a Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ at a resolution r that balances map fidelity against computational load. Given an initial pose estimate $\mathbf{p}_0 = [x_0, y_0, z_0, \phi_0, \theta_0, \psi_0]^T$ (from the previous fused pose), the algorithm refines this estimate by minimizing the negative log-likelihood of the transformed scan points under their enclosing Gaussians (more detail Section 3.2.1):

$$J(\mathbf{p}) = - \sum_{k=1}^N \log[\mathcal{N}(T(\mathbf{p}, \mathbf{x}_k) \mid \mu_{c(k)}, \Sigma_{c(k)})].$$

To meet real-time constrain, the computation of gradient(first derivative) and Hessian(second derivative) of the cost function are parallelized over T threads at each Newton iteration (up to $N_{\max}$). The set of LIDAR scan points $\{\mathbf{x}_k\}_{k=1}^N$ partitioned into disjoint subsets $\{\mathcal{K}_t\}_{t=1}^T$. Each thread t independently accumulates local gradient and Hessian contributions:

$$\mathbf{g}^{(t)} = \sum_{k \in \mathcal{K}_t} \nabla_p [-\log p(T(\mathbf{p}, \mathbf{x}_k))], \quad (4.5)$$

$$\mathbf{H}^{(t)} = \sum_{k \in \mathcal{K}_t} \nabla_p^2 [-\log p(T(\mathbf{p}, \mathbf{x}_k))]. \quad (4.6)$$

After all threads complete, a final reduction merges these into the global quantities. This lock-free design uses per-thread buffers and OpenMP’s guided scheduling to balance uneven voxel densities.

$$\mathbf{g} = \sum_{t=1}^T \mathbf{g}^{(t)}, \quad \mathbf{H} = \sum_{t=1}^T \mathbf{H}^{(t)}. \quad (4.7)$$

At each iteration the pose update uses a line-search-scaled Newton step:

$$\mathbf{p} \leftarrow \mathbf{p} - \alpha \mathbf{H}(\mathbf{p})^{-1} \mathbf{g}(\mathbf{p}), \quad \alpha \in [0, 1], \quad \|\Delta \mathbf{p}\| < \varepsilon. \quad (4.8)$$

The key parameters governing the NDT-OMP scan matching process are summarized in Table 4.1. These parameters control the resolution of the voxel grid, convergence criteria, optimization behavior, and parallelization strategy.

Table 4.1: Key parameters of the NDT-OMP Scan-to-Map Matching implementation

Parameter	Symbol	Description
Voxel resolution	r	Side length of each map voxel
Initial step size	α_0	Starting multiplier for Newton update
Convergence threshold	ε	Minimum norm of pose update for convergence
Maximum iterations	$N_{\max}$	Upper limit on Newton-method iterations
Thread count	T	Number of parallel threads used in optimization

Once scan matching converges, the resulting optimized pose is used as a global correction in the fusion stage, where it is integrated with LiDAR-inertial odometry using a factor graph-based optimization framework.

4.4 Real-Time Factor Graph Optimization

To fuse odometry from FAST-LIO2 with global corrections from scan-to-map matching, a factor graph with fixed-lag smoothing is employed. An Extended Kalman Filter (EKF) was initially considered, using FAST-LIO2 as the prediction model and map-based updates. However, the high-dimensional state of FAST-LIO2, which includes LiDAR-IMU calibration and bias terms, made EKF integration unstable and impractical.

The factor graph approach offers a more modular and scalable solution by modeling each constraint independently and optimizing them jointly. The use of a sliding window limits computation to recent states, ensuring real-time performance.

The following section details the formulation of the factor graph and the types of factors used.

4.4.1 Factor Graph Formulation

We denote the robot trajectory as a sequence of poses:

$$\mathcal{X} = \mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k, \quad \mathbf{x}_i \in SE(3) \quad (4.9)$$

where each pose $\mathbf{x}_i$ represents the 6-DOF position and orientation of the robot at discrete time step i , and $SE(3)$ denotes the special Euclidean group representing 3D rigid body transformations.

1. Prior Factor A prior factor is used to anchor the initial pose to a known global reference frame. It models the prior distribution over the initial state, expressed as:

$$\phi_{\text{prior}}(\mathbf{x}_0) = \exp\left(-\frac{1}{2}|\mathbf{x}_0 - \bar{\mathbf{x}}_0|^2 \Sigma_0\right) \quad (4.10)$$

In this formulation, $\mathbf{x}_0$ is the estimated initial pose of the robot. The term $\bar{\mathbf{x}}_0$ denotes the prior mean pose from the first LIO output or initialization, and Σ_0 is the prior covariance matrix reflecting the uncertainty. This Gaussian prior serves as a fixed anchor to stabilize the trajectory estimation.

2. Odometry (Between) Factors Relative motion from LIO is encoded using between factors, which represent the relative transformation between consecutive poses:

$$\phi_{\text{odom}}(\mathbf{x}_{i-1}, \mathbf{x}_i) = \exp\left(-\frac{1}{2}|f_{\text{odom}}(\mathbf{x}_{i-1}, \mathbf{x}_i) - \bar{\mathbf{z}}_i^{\text{odom}}|^2 \Sigma_{\text{odom}}\right) \quad (4.11)$$

Here, $\bar{\mathbf{z}}_i^{\text{odom}}$ is the relative pose measurement between time steps i and $i+1$, derived from the LIO module. The $f_{\text{odom}}(\cdot)$ function calculates the predicted relative transformation between the estimated poses $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$. The Σ_{odom} term is the associated covariance matrix, which quantifies the uncertainty in the odometry measurement. These between factors form the sequential backbone of the trajectory by chaining consecutive pose estimates together through relative motion constraints.

3. Map-Matching Priors Absolute pose estimates derived from scan-to-map matching using Normal Distributions Transform (NDT) are incorporated as prior constraints on select poses:

$$\phi_{\text{map}}(\mathbf{x}_j) = \exp \left(-\frac{1}{2} |\mathbf{x}_j - \bar{\mathbf{x}}_j^{\text{map}}|^2 \Sigma_{\text{map}} \right) \quad (4.12)$$

In this expression, $\bar{\mathbf{x}}_j^{\text{map}}$ denotes the global pose obtained by aligning the current LiDAR scan with a pre-built prior map using scan matching techniques. The associated covariance matrix Σ_{map} expresses the uncertainty of the registration process, accounting for alignment quality and sensor noise. These priors provide absolute spatial constraints within the factor graph, which help to correct accumulated drift from relative odometry and enhance the global consistency of the trajectory estimation.

4.4.2 Maximum A Posteriori (MAP) Estimation

The objective of the sensor fusion system is to compute the most probable robot trajectory given all available measurements, including odometry and map-matching observations. This is formulated as a Maximum A Posteriori (MAP) estimation problem, expressed as the following nonlinear least-squares objective:

$$\mathbf{X}^* = \arg \min_{\mathbf{X}} \left(\|\mathbf{x}_0 - \bar{\mathbf{x}}_0\|_{\Sigma_0}^2 + \sum_{i=1}^k \|f_{\text{odom}}(\mathbf{x}_{i-1}, \mathbf{x}_i) - \bar{\mathbf{z}}_i^{\text{odom}}\|_{\Sigma_{\text{odom}}}^2 + \sum_{j \in \mathcal{M}} \|\mathbf{x}_j - \bar{\mathbf{x}}_j^{\text{map}}\|_{\Sigma_{\text{map}}}^2 \right) \quad (4.13)$$

Each term in the cost function represents a Mahalanobis distance, penalizing deviations from the respective measurements weighted by their uncertainty. Solving this objective yields the optimal trajectory estimate $\mathbf{X}^*$, which fuses relative odometry and global pose observations in a probabilistically consistent manner.

4.4.3 Fixed-Lag Smoothing and Marginalization

In our proposed system, fixed-lag smoothing is used to maintain real-time performance while optimizing the recent trajectory of the robot. Instead of keeping the entire history of poses in the factor graph, we define a sliding time window of length τ , such that only poses within the interval $[t_k - \tau, t_k]$ are retained, where t_k is the current time. This ensures that the graph grows only within a fixed temporal horizon, regardless of how long the system has been running.

Let $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k$ represent the sequence of robot poses over time. At each time step, the latest pose $\mathbf{x}_k$ is added to the factor graph along with new factors derived from LIO

(odometry) and map-matching. Older poses such as $\mathbf{x}_i$ for which $t_i < t_k - \tau$ are marginalized out to limit the graph size.

The process of marginalization involves removing a variable (pose) from the graph while preserving its influence on the remaining variables. Suppose the state vector is partitioned as:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_m \\ \mathbf{x}_r \end{bmatrix}, \quad (4.14)$$

where $\mathbf{x}_m$ are the variables to be marginalized (e.g., old poses outside the lag window), and $\mathbf{x}_r$ are the variables retained in the active window. The joint probability distribution is expressed as:

$$P(\mathbf{x}_m, \mathbf{x}_r) = P(\mathbf{x}_m | \mathbf{x}_r) P(\mathbf{x}_r). \quad (4.15)$$

To obtain a reduced representation, we marginalize out $\mathbf{x}_m$:

$$P(\mathbf{x}_r) = \int P(\mathbf{x}_m, \mathbf{x}_r) d\mathbf{x}_m. \quad (4.16)$$

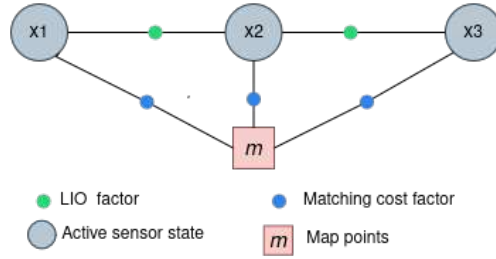
As a result, a new dense factor is introduced between variables in $\mathbf{x}_r$ that were connected to $\mathbf{x}_m$, preserving the influence of the eliminated variables.

The process of fixed-lag smoothing is illustrated in Figure 4.3 using a temporal window of length $n = 4$. Initially (Figure 4.3a), the system contains states $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$, connected by odometry and map-matching factors. As pose $\mathbf{x}_4$ is added at time t_4 (Figure 4.3b), it links to $\mathbf{x}_3$ through a new odometry factor and includes a map constraint. Since all poses $\mathbf{x}_1$ to $\mathbf{x}_4$ lie within the lag window $[t_1, t_4]$, the graph remains fully active and no marginalization occurs.

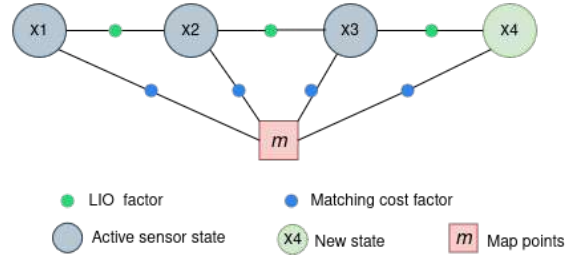
When the system reaches time step t_5 , pose $\mathbf{x}_5$ is added, along with its corresponding odometry and map-matching constraints. At this point, pose $\mathbf{x}_1$ falls outside the fixed-lag window $[t_2, t_5]$. As a result, it is marginalized. This process eliminates $\mathbf{x}_1$ from the graph but replaces its influence with a dense prior that connects to the remaining relevant states, particularly $\mathbf{x}_2$ and beyond (Figure 4.3c). The marginalized node is no longer directly optimized, but its effect is preserved through conditional probability in the form of a Bayes net.

At the next step, time t_6 , a new pose $\mathbf{x}_6$ is introduced. To maintain the lag window $[t_3, t_6]$, the system marginalizes pose $\mathbf{x}_2$, introducing another dense factor that connects to the active variables. The current active states now consist of $\mathbf{x}_3$, $\mathbf{x}_4$, $\mathbf{x}_5$, and $\mathbf{x}_6$, all within the sliding lag window (Figure 4.3d).

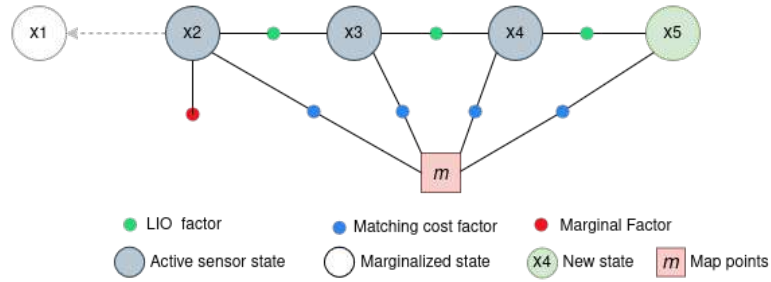
This incremental update mechanism, implemented using the GTSAM library, maintains a bounded graph size by marginalizing old variables into Bayes net conditionals while pre-



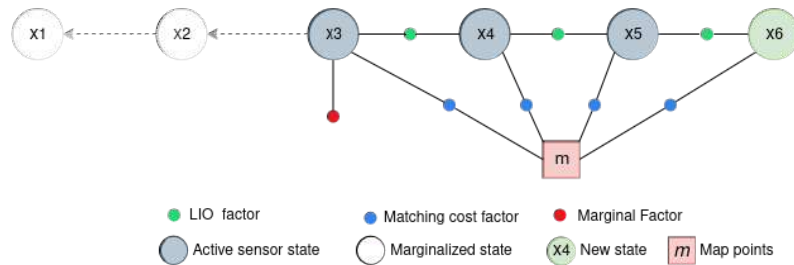
(a) The initial factor graph contains active states $\mathbf{x}_1$ to $\mathbf{x}_3$, linked through LiDAR-inertial odometry (LIO) and map-matching factors.



(b) A new state $\mathbf{x}_4$ is introduced, remaining within the lag window.



(c) As state $\mathbf{x}_5$ is added, the oldest state $\mathbf{x}_1$ is marginalized and its influence is preserved via a prior factor.



(d) At time t_6 , $\mathbf{x}_6$ is introduced and $\mathbf{x}_2$ is marginalized, maintaining a fixed-lag window over the most recent four states.

Figure 4.3: Demonstration of fixed-lag smoothing with a lag size of $n = 4$ in a factor graph-based localization system.

serving historical information. Recent poses within the active window are kept in a factor graph and updated via iSAM2, striking a balance between computational efficiency and estimation accuracy for real-time localization.

4.4.4 Outlier Rejection via Mahalanobis Distance

To ensure robust fusion of odometry and scan-map correction constraints, we apply a probabilistic outlier test based on the Mahalanobis distance. Let the correction measurement be the 6-DOF pose $\mathbf{z} = [x, y, z, \phi, \theta, \psi]^T$ provided by the NDT-based scan matching, with associated covariance Σ_z . Given the current filter estimate $\hat{\mathbf{z}}$, we compute the innovation

$$\Delta \mathbf{z} = \mathbf{z} - \hat{\mathbf{z}}.$$

The squared Mahalanobis distance is then

$$D_M^2 = \Delta \mathbf{z}^T \Sigma_z^{-1} \Delta \mathbf{z}. \quad (4.17)$$

Under the Gaussian assumption, D_M^2 follows a χ^2 distribution with $d = 6$ degrees of freedom. We reject the correction if

$$D_M^2 > \chi_{d,0.95}^2, \quad (4.18)$$

where $\chi_{d,0.95}^2$ is the 95% quantile of the χ_d^2 distribution. Otherwise, the measurement is accepted and incorporated into the factor graph. This criterion effectively discards implausible scan-matching corrections while preserving valid global pose updates.

4.5 Dynamic Object Detection and Removal

4.5.1 Dynamic Object Detection

To identify and remove dynamic objects from the LiDAR point cloud, this work utilizes a deep learning-based 3D object detection pipeline based on CenterPoint[27]. The model is developed and trained using the MMDetection3D framework[38], and is deployed for real-time inference using the Autoware-compatible implementation[39].

The detection architecture adopts the two-stage design of CenterPoint, as shown in Figure 3.2. First, the raw LiDAR point cloud is voxelized and processed by a voxel feature encoder and a PointPillars-based sparse convolutional backbone[28], which extracts high-level spatial features. These features are projected into a Bird's-Eye View (BEV) map, enabling efficient object detection. A keypoint-based detection head then identifies object centers and regresses 3D bounding box attributes, including size, orientation, and velocity.

The final output includes class-labeled 3D detections, which are used to support dynamic object removal.

Instead of training from scratch, this work utilizes a pretrained CenterPoint model trained on the nuScenes dataset [30], comprising over 28k LiDAR frames. The model includes five object classes: CAR, TRUCK, BUS, BICYCLE, and PEDESTRIAN. These categories cover the primary dynamic agents present in urban driving scenarios. After training, the model is exported to the ONNX (Open Neural Network Exchange) format, which supports cross-platform deployment and enables hardware-accelerated inference using TensorRT.

4.5.2 Detected Object removal

Following object detection, dynamic elements are removed from the LiDAR point cloud using their predicted 3D bounding boxes. For each detected object, its position, dimensions, and orientation are used to define an oriented 3D cropping volume. These bounding boxes are expanded slightly with a fixed margin to ensure full exclusion of the object’s points, accounting for detection uncertainty or sensor noise.

Each bounding box is then used to define a geometric filter in 3D space. The removal process involves identifying all LiDAR points that fall within the transformed bounding box and excluding them from the original scan. This is performed iteratively for all detected dynamic objects in each frame. The result is a filtered point cloud that retains only static environmental features, which are then used for downstream localization tasks.

4.6 Environmental Noise Simulation

To assess the robustness of the proposed localization pipeline under adverse conditions, we simulate the impact of atmospheric fog on LiDAR scans. Fog causes scattering and absorption of laser beams, leading to partial loss of valid returns, the appearance of false near-field echoes, and a reduction in return intensity. Our simulation adopts the physically grounded model proposed by Teufel et al. [40], applied only to live LiDAR scans. The degradation severity is controlled by the *meteorological visibility* V (in meters), defined as the maximum distance at which an object can be reliably seen under foggy conditions.

The fog model introduces three major effects, each governed by simple visibility-dependent equations: (i) deletion of real points, (ii) modification into near-sensor backscatter points, and (iii) intensity attenuation.

Point Deletion. Real LiDAR returns are probabilistically removed based on their distance d and visibility V , using:

$$p_{\text{delete}}(d) = \frac{a \cdot e^{bV}}{a \cdot e^{bV} + 1}$$

Here, $a = -0.70$ and $b = -0.024$ are empirically derived constants that define the effect of fog density on the likelihood of return loss. Deletion probability increases with distance and with denser fog (lower V).

Backscatter Modification. Points that are not deleted may be converted into *false returns*, simulating backscatter from fog droplets. The probability of modification increases with distance:

$$p_{\text{modify}}(d) = 1 - e^{-d \cdot \epsilon}, \quad \epsilon = 0.23 \cdot e^{-0.0082V}$$

The new range is drawn from an exponential distribution:

$$f(x) = \frac{1}{\lambda} e^{-x/\lambda}, \quad \lambda = -0.006 \cdot V + 2.31$$

This causes the point to be radially pulled closer to the sensor, mimicking typical fog backscatter effects.

Intensity Attenuation. Return intensity is reduced due to scattering loss along the beam path:

$$I(d) = I_0 \cdot e^{-\gamma d}, \quad \gamma = 0.02$$

Points with final intensity below a threshold (e.g., 1000 for 16-bit returns) are discarded. Simulated backscatter points are assigned random intensities in the range of 0–32% of the maximum value, consistent with real sensor behavior under fog.

This fog simulation enables a physically interpretable, parameterized degradation model suitable for evaluating localization performance under various visibility conditions.

Chapter 5

Experimental Setup and Result

This chapter presents the experimental framework and evaluation of the proposed LiDAR-inertial localization system. It includes the hardware and software configurations, dataset preparation, and a detailed performance analysis under both standard and challenging conditions. The evaluation focuses on accuracy, robustness, and real-time performance across different environments.

5.1 Experimental Environment

5.1.1 Hardware and Software Setup

All experiments were conducted on a development laptop equipped with an AMD Ryzen™ 7 processor, 32 GB of DDR5 RAM, and NVIDIA GPU for accelerated computation, as well as on a Hunter mobile robotic platform for on-campus data collection. The robot is equipped with an Ouster OS1-128 LiDAR sensor for 3D environmental perception, an on-board Inertial Measurement Unit (IMU) for motion estimation, and a camera for visual reference (Figure 5.1).

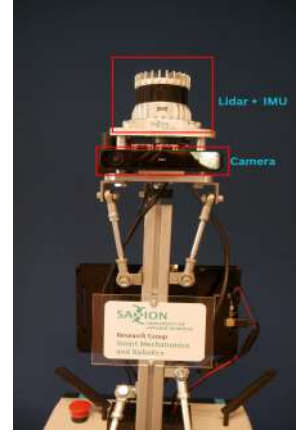
Our approach is implemented primarily in C++ on top of the ROS 2 Humble middleware. We rely on:

- **PCL (Point Cloud Library):** for point cloud data structures and filtering.
- **Eigen:** for linear algebra operations (transformations, matrix computations).

Additional scripts in Python were employed for plotting and minor data manipulations.



(a) Hunter mobile robot platform



(b) Sensor configuration (LiDAR, IMU, Camera)

Figure 5.1: Hardware setup used for real-world data collection and testing: (a) Hunter robotic platform; (b) onboard sensors.

5.1.2 Datasets Collection and Map Preparation

We evaluate the proposed localization system across a combination of public benchmark datasets and custom-recorded sequences. These datasets include suburban, campus, and indoor environments with varying levels of dynamic activity, structural repetition, and trajectory complexity. Table 5.1 summarizes the key characteristics of each sequence, including the environment type and total trajectory length.

Table 5.1: Dataset Overview and Environmental Characteristics

Dataset (Seq ID)	Environment Description	Trajectory
Saxion Seq-01	Outdoor campus area with moderate pedestrian activity and smooth motion.	~0.9 km/12 min
Saxion Seq-02	Outdoor campus area with dynamic objects (pedestrians, parked cars), sharp turns, low loop closure opportunities and narrow passages.	~1.15 km/16 min
Saxion Seq-03	Long sub-urban path with dynamic objects (pedestrians, parked cars), mixed building heights and repetitive environment (apartment blocks).	~1.8 km/27 min
Saxion Seq-04	Indoor area characterized by long glass-walled corridors and structurally repetitive office layouts.	~0.43 km/11 min
KITTI Seq-05	Residential environment with urban roads, sidewalks, parked vehicles, repeating patterns, tree, moderate traffic and fast motion.	~2.2 km/5 min
MulRan KAIST-03	campus area with moderate traffic and multiple distinguishable structures	~6.2 km/14 min

Benchmark Dataset We used sequences 05 from the KITTI odometry benchmark[41] and sequences KAIST-03 from the MulRan dataset[42]. The official ground-truth poses were used to transform and aggregate each LiDAR frame into a global coordinate system for generate 3D priori map.

Custom Saxion Dataset As no direct ground truth was available for this dataset, we employed Fast-LIO2 for initial SLAM-based mapping, followed by offline global optimization using Hierarchical LiDAR Bundle Adjustment (HBA) [43] to ensure global consistency across repeated traversals.

In both cases, the final global point cloud is subdivided into manageable tiles for efficient local map loading during localization. Each tile is stored as a separate file or database entry keyed by tile coordinates, allowing the system to load only relevant tiles based on the vehicle’s current estimated position.

5.2 Experimental Evaluation

5.2.1 Evaluation Metrics and Methodology

To assess the localization performance, we use a combination of translational and rotational error metrics, with a focus on Absolute Pose Error (APE). APE quantifies the Euclidean distance between the estimated trajectory and the reference trajectory after applying full SE(3) alignment using the Umeyama method. This alignment ensures fair comparison by removing global offsets in rotation, translation, and scale.

All trajectory evaluations were conducted using the EVO evaluation toolkit [44], which provides statistical metrics including root mean square error (RMSE), mean, median, and standard deviation. For each sequence, we align the estimated trajectory to the ground truth and report the APE over time, along with summary statistics.

In addition to APE, we assess the system’s real-time performance by measuring:

- Per-frame execution time across key modules (preprocessing, scan matching, optimization),
- Registration convergence rate and iteration count,
- Localization success rate under environmental degradation (e.g., fog, sparse features).

The evaluation includes comparisons against two categories of : component baselines, which consist of the individual modules integrated into our system (Fast-LIO2[13] and scan-map-matching scan matching), and external map-based localization methods.

5.2.2 Localization Performance

Comparison with baseline methods

Quantitative results for the Saxion sequences are presented in Tables 5.2, 5.3, and 5.4, summarizing translational and rotational error statistics (RMSE, mean, and maximum) after SE(3) alignment. Each sequence represents a different environmental condition and trajectory complexity. The results show that the proposed fusion-based system consistently outperforms both baseline methods across all sequences.

In Sequence 1 (see Table 5.2), the proposed method achieves centimeter-level accuracy, with translation RMSE of 0.067 m, compared to 5.47 m in Fast-LIO2. The rotational RMSE is also reduced from 2.072° to 0.583° , indicating significant drift correction not only in position but also in orientation.

Table 5.2: Translation (APE) and Rotation Error statistics for Saxion **Sequence 1**

Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed (Fusion)	APE (m)	0.395	0.052	0.041	0.003	0.067	0.042
	Rot. (deg)	4.715	0.321	0.169	0.030	0.583	0.486
NDT Map Matching	APE (m)	0.363	0.074	0.067	0.012	0.084	0.041
	Rot. (deg)	5.832	0.503	0.299	0.012	0.813	0.639
Fast-LIO2	APE (m)	8.94	4.456	4.933	0.322	5.47	1.645
	Rot. (deg)	6.213	1.631	1.304	0.419	2.072	1.278

Note: Bold values indicate the best performance across each metric.

In more complex scenarios (Saxion Sequences 2 and 3), which include dynamic agents, longer sequence, and sharp turns, the proposed fusion method achieves decimeter-level accuracy (see Figure 5.3), significantly reducing translational RMSE compared to Fast-LIO2 (from 3.8 m to 0.11–0.13 m) and improving rotational RMSE by up to 1.6° . While the proposed method exhibits a slightly increase its error in some cases, it consistently achieves lower RMSE, indicating better average accuracy and more stable performance across the entire trajectory.

Across all sequences, the APE translation standard deviation remains within a narrow range of 0.042–0.093 meters and APE rotation 0.4 - 0.74 Degree, indicating that the proposed system achieves both low error and high temporal consistency.

Trajectory plots Figures 5.2, A.1, and A.2 illustrate the alignment of estimated poses with reference trajectory. Zoomed-in regions further emphasize the proposed method’s ability to track the trajectory accurately, even in complex zones and long trajectory.

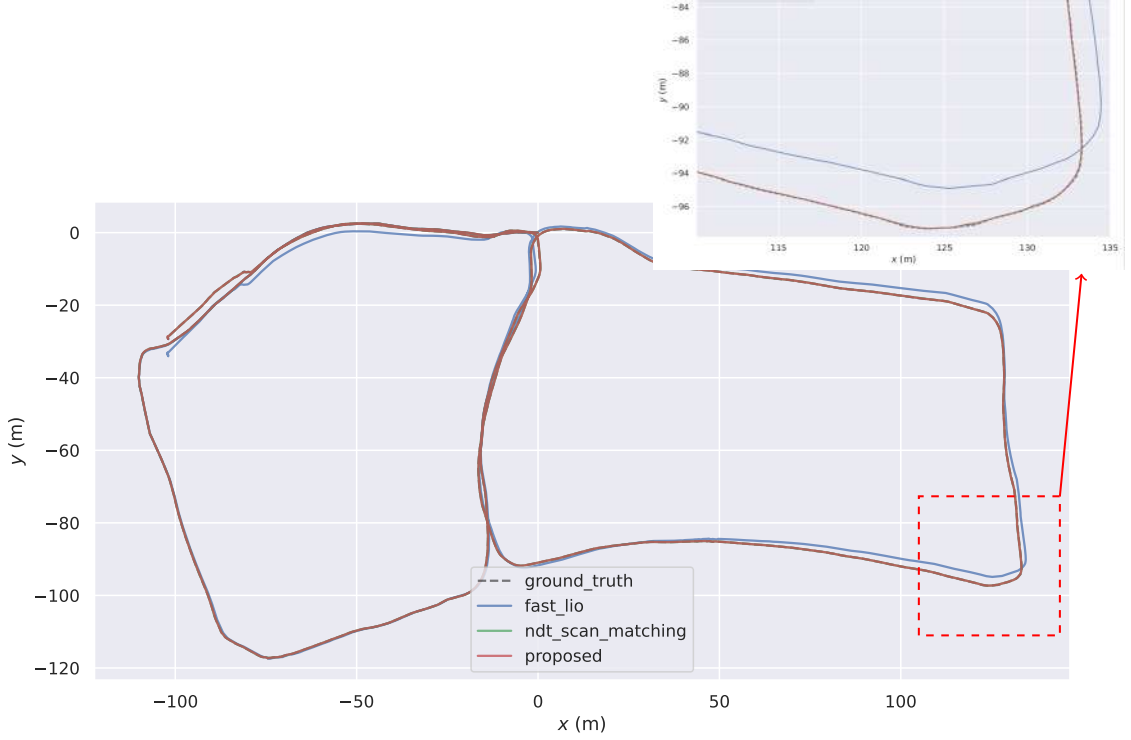


Figure 5.2: **Saxion Sequence 01 – Trajectory Alignment.** The figure shows the estimated trajectories from FAST-LIO2, low frequency NDT map matching , and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region.

Table 5.3: Translation (APE) and Rotation Error statistics for Saxion **Sequence 2**

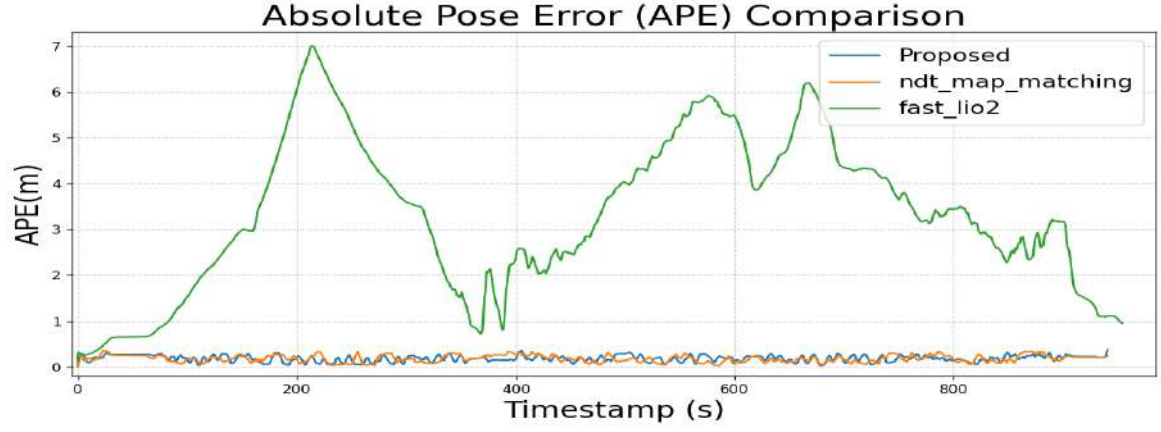
Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed (Fusion)	APE (m)	0.440	0.090	0.075	0.005	0.107	0.058
	Rot. (deg)	4.356	0.782	0.539	0.051	1.046	0.695
NDT Map Matching	APE (m)	0.245	0.102	0.094	0.009	0.118	0.059
	Rot. (deg)	2.639	1.195	1.127	0.103	1.393	0.715
Fast-LIO2	APE (m)	6.9941	3.5576	3.44	0.070	3.88	3.8813
	Rot. (deg)	5.494	2.250	1.568	0.202	2.680	1.456

Note: Bold values indicate the best performance across each metric.

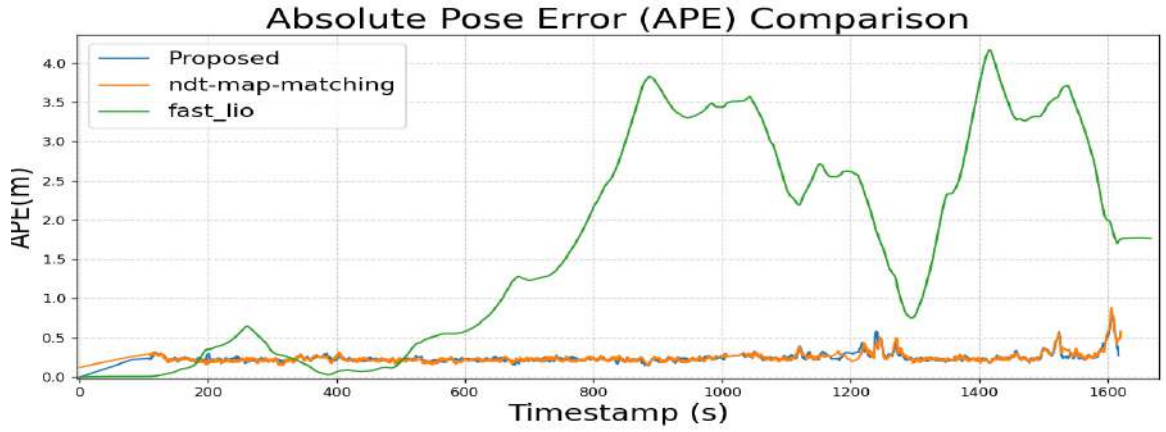
Table 5.4: Translation (APE) and Rotation Error statistics for Saxion **Sequence 3**

Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed (Fusion)	APE (m)	0.821	0.097	0.072	0.004	0.135	0.094
	Rot. (deg)	2.016	0.725	0.472	0.076	1.039	0.743
NDT Map Matching	APE (m)	0.297	0.113	0.100	0.025	0.126	0.056
	Rot. (deg)	3.236	1.373	1.286	0.131	1.652	0.919
Fast-LIO2	APE (m)	4.075	1.468	1.254	0.028	2.162	1.145
	Rot. (deg)	2.750	0.955	0.953	0.069	1.088	0.522

Note: Bold values indicate the best performance across each metric.



(a) APE Saxion Sequence 2



(b) APE Saxion Sequence 3

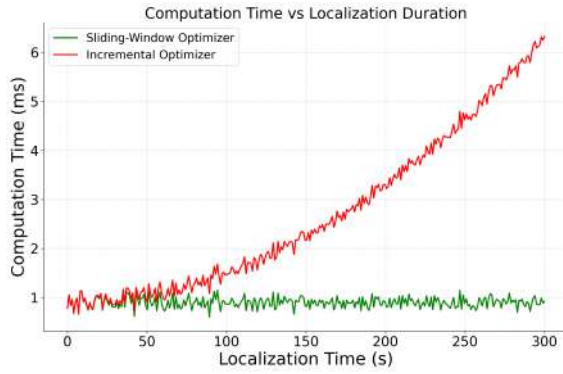
Figure 5.3: Absolute Pose Error Comparison of (a) Saxion Sequence 2 and (b) Saxion Sequence 3 .

Real-time Performance

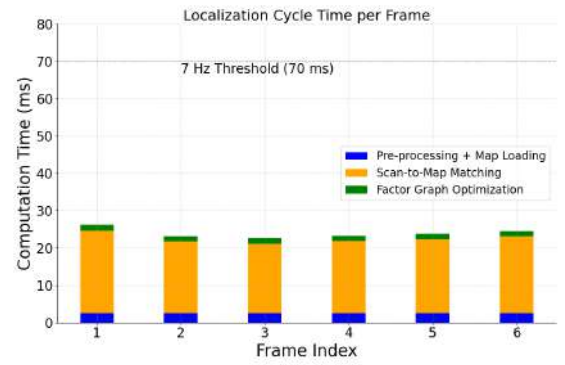
We evaluate scan-to-map matching by varying the radius of the local sub map and compare multi-threaded NDT (NDT-OMP) , standard NDT and classic ICP (Figure 5.5). As shown in the Table 5.5 While both NDT and ICP maintain convergence on small radii (> 100 m), their per-scan runtime is exceed 100 ms and increases as map grows. In contrast, NDT-OMP maintains sub-20 ms mean latency with over 95 % convergence up to 200 m, and still achieves under 25 ms runtimes and 90 % convergence at 350 m (with occasional failures). This demonstrates that NDT-OMP's parallelization, coupled with an adaptive local-map radius based on the robot's pose, significantly improves scan-matching efficiency without sacrificing robustness.

Table 5.5: Scan-Matching Performance vs. Local Map Radius

Map Radius(m)	Method	Mean Time(ms)	Converged(%)	Remarks
100	NDT	90	90	
	ICP	>100	90	
	NDT-OMP(8 thread)	15	99	
200	NDT	>100	<50	failures
	ICP	>100	<50	failures
	NDT-OMP(8 thread)	20	97	
350	NDT	>100	<50	failures
	ICP	>100	<50	failures
	NDT-OMP(8 thread)	25	90	some failures



(a) Computation time trend of sliding-window vs full-batch optimizer.



(b) Computation time breakdown per frame (stacked components).

Figure 5.4: Computation characteristics of the proposed localization system: (a) Sliding-window optimization scalability and (b) real-time frame-wise timing breakdown.



Figure 5.5: Top-down view of the localization trajectory overlaid on the pre-built 3D map.

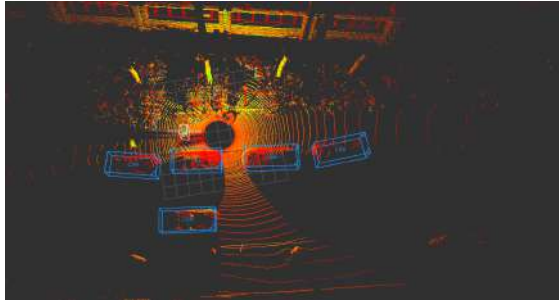
The red line indicates the estimated trajectory, while the white rectangle structures represent dynamically loaded local submaps within a fixed radius during runtime. This demonstrates how local map tiling supports scalable scan-to-map registration.

Considering a high-frequency LiDAR-inertial odometry (LIO) stream and periodic map-matching updates, we compare a sliding-window factor-graph optimizer against a full-batch solver. By restricting the graph to the most recent 5–50 seconds of data, as shown in Figure 5.4a the sliding-window approach maintains nearly constant solve times under 1 ms per update regardless of localization duration. In contrast, the full-batch solver’s complexity grows superlinearly as more keyframes accumulate, eventually fail in real-time as the number of nodes increase.

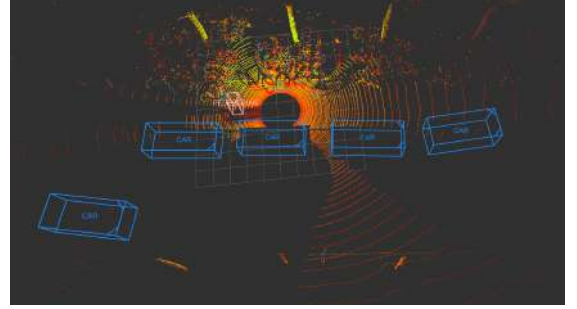
Figure 5.4b further breaks down the per-frame latency of the proposed system. With point-cloud pre-processing and map loading taking 2 ms, scan-to-map matching 20 ms, and factor-graph optimization 1 ms, the total remains below 23 ms. This comfortably satisfies the real-time 10 Hz requirement (100 ms), achieving 43 Hz processing frequency.

Impact of Dynamic Object Removal on Registration Performance

In this evaluation, we assess the impact of incorporating dynamic object removal into the localization pipeline. Example of detected 3D bounding boxes and the corresponding retained point cloud after dynamic object removal are illustrated in Figure 5.6. The objective is to analyze how removing transient objects from LiDAR scans influences registration accuracy, localization robustness, and computational efficiency. Key performance indicators include registration convergence rate, average iteration count and execution time are evaluated.



(a) 3D bounding box of detected dynamic objects



(b) Point-cloud retained after removing points with 3d bounding box.

Figure 5.6: Detecting and removing dynamic objects from 3d LIDAR point-cloud.

Table 5.6: Comparison of Point-cloud Registration Metrics Before and After Dynamic Object Removal

Metric	Without Removal	With Removal	Improvement
Avg. Registration Iterations	19	12	↓ 7
Avg. Execution Time (ms)	14.41	9.73	↓ 4.68
Registration Failures	7	2	↓ 5

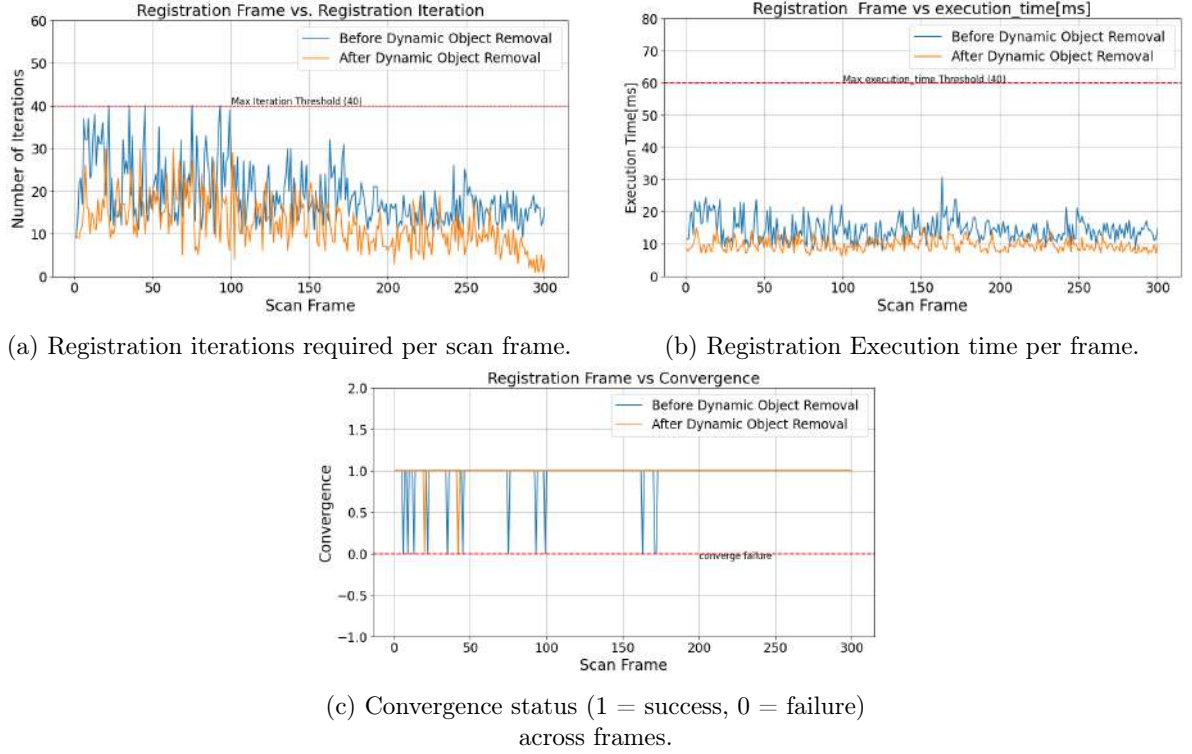


Figure 5.7: **Registration performance before and after dynamic object removal.**

Each subplot shows the impact of removing dynamic objects on and iteration count (a) , execution time (b), and convergence (c). Across 300 frames, the system becomes more stable, efficient, and reliable after filtering out dynamic elements from the scan.

The point-cloud registration performance was evaluated across 300 scan frames , selected based on a high number of detected dynamic objects. As summarized in Table 5.6, filtering dynamic objects results reduction in average registration iterations (from 19 to 12) and execution time (from 14.41 ms to 9.73 ms). Additionally, the number of convergence failures dropped from 7 to 2. The performance plots in Figure 5.7 further illustrate this trend. Dynamic object removal improves stability and efficiency by eliminating noisy inputs that otherwise increase iteration count, processing time, or lead to failed alignments.

Table 5.7 presents the computational overhead introduced by integrating dynamic object detection and filtering into the localization pipeline. While this module enhances registration robustness and accuracy, it incurs additional processing time. Specifically, the 3D object detection step (using ONNX inference) adds approximately 28 ms, and the point cloud filtering (using a CropBox filter) adds another 3 ms per frame. Although the registration and optimization stage benefits from a reduction in processing time from 15 ms to 10 ms the overall per-frame runtime increases from 15 ms (without removal) to 41 ms (with removal), resulting in a net overhead of 26 ms. Despite this increase, the system remains within acceptable real-time performance bounds for medium-speed mobile robotics applications.

Table 5.7: Per-frame runtime analysis with and without dynamic object removal.

Component	Without Removal (ms)	With Removal (ms)	Overhead(ms)
Object Detection	—	28.0	+28.0
Point Cloud Filtering	—	3.0	+3.0
Registration and other	15.0	10.0	↓ 5
Total	15.0	41.0	+26.0

Localization on Map Side Feature Sparse Environment

Map-side feature-sparse environments refer to areas where the current LiDAR observations contain few or no correspondences in the prebuilt map. This situation typically arises at the boundaries of a mapped area, in unmapped zones, or at transitions between separately built or merged submaps. In such regions, the prior map lacks persistent geometric features required for reliable scan-to-map registration, resulting in degraded or failed localization if scan matching is used in isolation.



Figure 5.8: **Localization in transition zone.** The robot enters a transition region (highlighted in red) where LiDAR scan matching fails due to the absence of stable features in the prebuilt map

In this scenario, the robot traverses an approximately 90-meter-long transition corridor connecting two main streets. This zone, highlighted in red in Figure 5.8, contains both partially mapped and unmapped segments. The traversal was performed twice to assess consistency. As shown in Figure 5.11, the NDT-based map matching exhibited two prominent error spikes (each exceeding 2 m) corresponding precisely to the robot’s entry into the unmapped region during both passes. These spikes highlight NDT’s reliance on strong map overlap and its vulnerability in feature-deprived zones.

In contrast, the proposed fusion pipeline maintained stable localization throughout both traversals, as shown in Figure 5.11. This stability is attributed to the FAST-LIO front-end, which provides high-frequency odometry estimates unaffected by the absence of map

data. By combining this with map-based corrections via factor graph optimization, the system successfully bridges the gap between locally consistent and globally drift-resilient localization.

Table 5.8 summarizes the APE statistics. The proposed method limited the maximum APE to 0.455 m, compared to 2.178 m with NDT. The RMSE was also significantly lower—0.095 m versus 0.295 m. These results demonstrate the robustness of the fusion approach under degraded mapping conditions.

For comparison, under fully mapped conditions (Saxion Sequence 1, Table 5.2), the same system achieved a maximum APE of 0.395 m and an RMSE of 0.067 m. This indicates that the proposed method maintains localization accuracy even when operating in feature-sparse or partially unmapped environments.

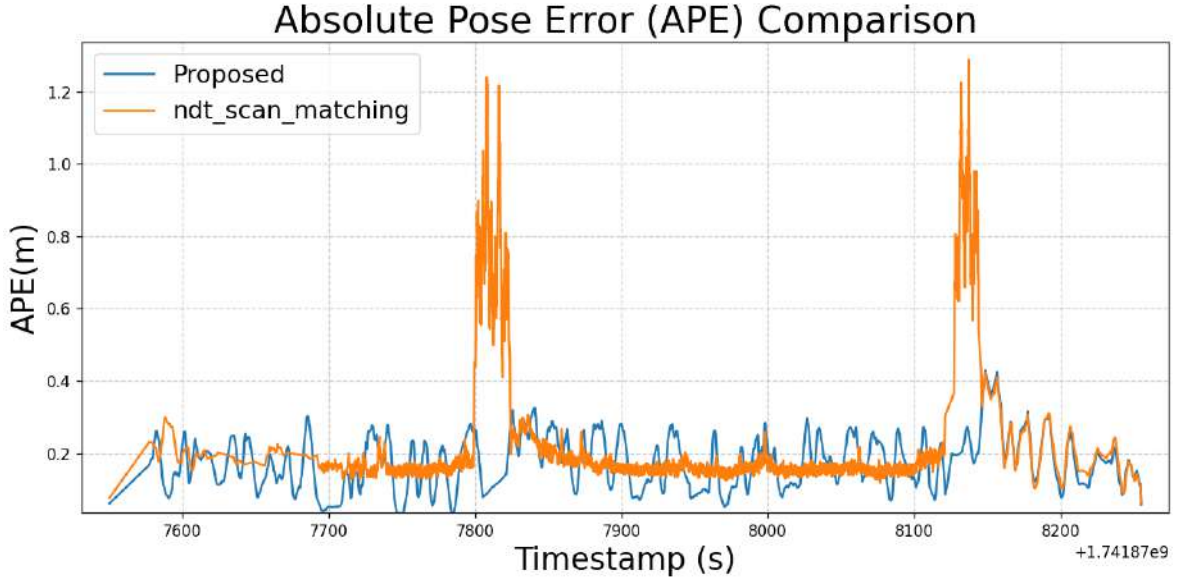


Figure 5.9: **Error comparison in transition zone.** The APE for NDT map matching shows two large spikes (a), while the proposed fusion approach remains stable (b), benefiting from the FAST-LIO front-end.

Table 5.8: Absolute Pose Error (APE) Statistics in Transition Corridor

Method	Max APE (m)	Mean APE (m)	RMSE (m)	Std Dev (m)
Proposed Fusion	0.455	0.078	0.095	0.054
NDT Map Matching	2.178	0.131	0.295	0.264
FAST-LIO2	7.59	1.456	2.197	1.645

Note: Bold values indicate the best performance across each metric.

Localization Under Fog Degradation with a Recent Map

We tested the localization pipeline on the **Saxion Sequence 1** using a one-month-old prebuilt map and simulated fog at three visibility levels: *Mild* (100 m), *Moderate* (60 m),

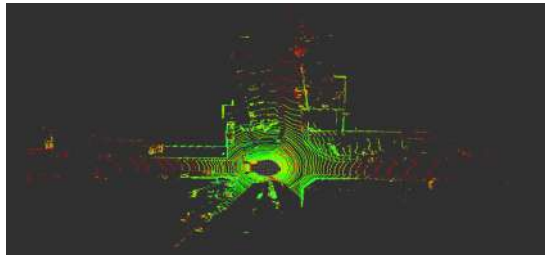
and *Severe* (30 m) see Table 5.9. Figure 5.10 illustrates fog-induced degradation effects such as point dropouts, false near-field returns, and spatial clutter. The resulting Absolute Pose Error (APE) was compared against a baseline under clear conditions.

Under Mild and Moderate fog Table 5.10, the proposed fusion and NDT map matching module methods show limited degradation compared to the baseline (Table 5.2). APE increases to 0.10-0.28 m, with occasional peaks over 1.4 m, but RMSE remains stable, confirms the robustness of NDT under moderate visibility losses. In contrast, FAST-LIO2 drifts significantly, with higher mean RMSE values, indicating its sensitivity to fog when operating without map-based correction.

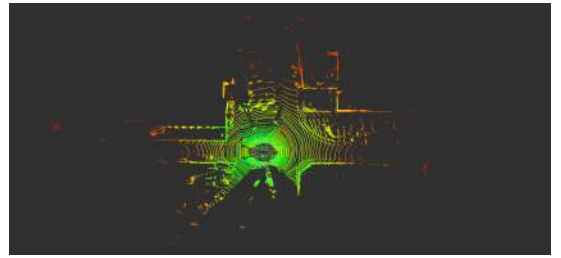
Under Severe fog (visibility 30 m), all methods show significant degradation. The NDT map matching module becomes unstable and fails to converge consistently due to severe point dropout and noise. Fast-LIO also exhibits large drift, and the combined effect causes the fusion pipeline to fail repeatedly, unable to maintain consistent localization beyond a short distance.

Table 5.9: Fog severity levels based on visibility range

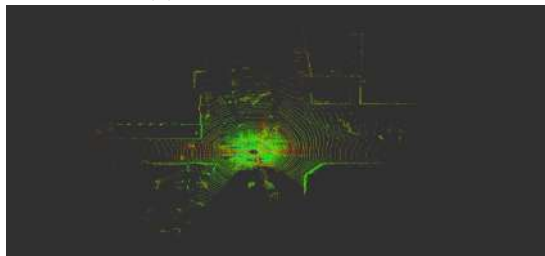
Fog Level	Visibility V (m)
Mild	100
Moderate	60
Severe	30



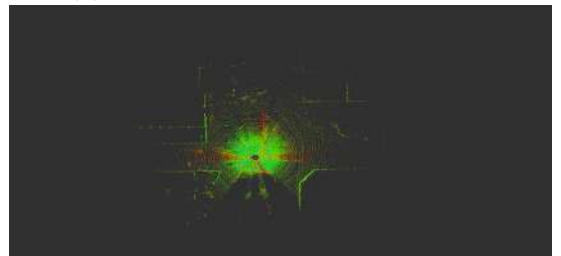
(a) Original Point-cloud



(b) Mild noise level added point-cloud



(c) Moderate noise level added point-cloud



(d) Severe noise level added point-cloud

Figure 5.10: Point-cloud snapshots under increasing composite geometric noise levels: (a) Original scan; (b) Mild noise; (c) Moderate noise; (d) Severe noise.

Table 5.10: Translation error (APE) statistics under Severe and Moderate fog visibility

Condition	Method	Max	Mean	Min	RMSE	Std Dev
Mild	Proposed	0.3682	0.1238	0.0250	0.1356	0.0583
	NDT Map Matching	0.4089	0.1474	0.0164	0.1585	0.0581
	Fast-LIO2	12.0253	4.2643	0.581	5.7521	3.2145
Moderate	Proposed	0.3792	0.1326	0.0092	0.1452	0.0593
	NDT Map Matching	1.3442	0.1712	0.0193	0.2178	0.1346
	Fast-LIO2	17.7823	6.23521	0.0610	12.9345	5.3614

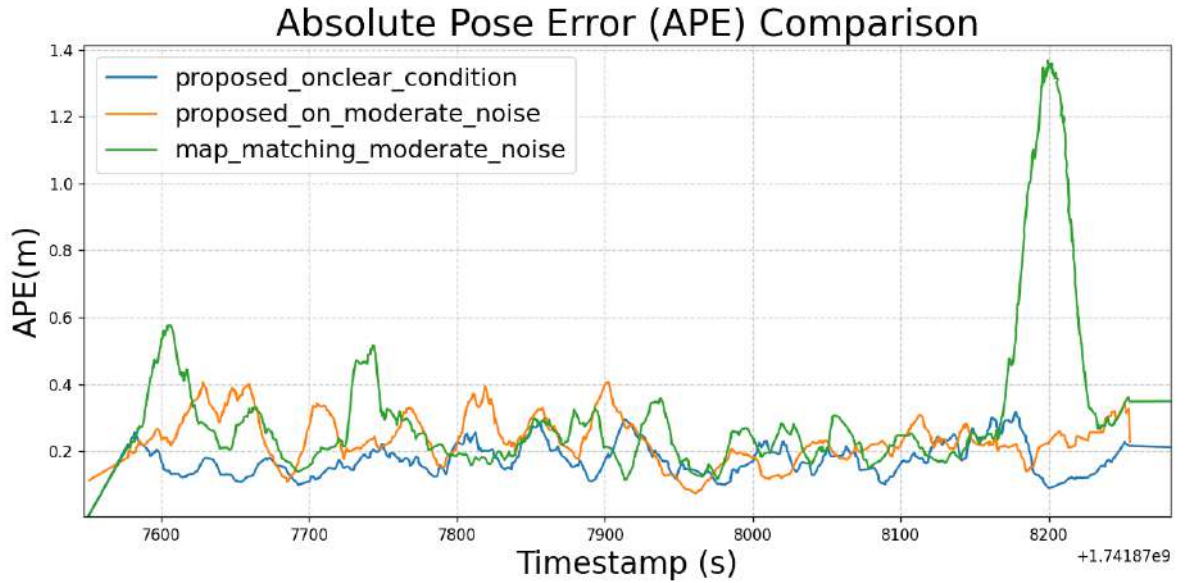


Figure 5.11: **Error compression in transition zone.** The APE for NDT map matching shows two large spikes (a), while the proposed fusion approach remains stable (b), benefiting from the FAST-LIO front-end.

Benchmarking on Public Dataset

To evaluate the effectiveness of the proposed method, we compare its mean translation and rotation errors against recent map-based localization techniques using the KITTI Sequence 05 dataset. The comparison includes methods based on stereo vision and prior maps [1], visual point cloud priors [45], and LiDAR-only map alignment [23]. Table 5.11 summarizes the results, where our approach demonstrates superior accuracy and consistency, with lower mean error and deviation across both translation and rotation metrics.

To highlight the benefits of using a prior map, we provide a contextual comparison with SLAM-based methods, which follow a fundamentally different paradigm. While SLAM systems rely on loop closures and global optimization to reduce drift, this is often insufficient to fully correct long-term error—as seen in KITTI Sequence 05. In contrast, our method

Table 5.11: Comparison of translation and rotation errors (mean and standard deviation) on KITTI Sequence 05.

Method	Translation Error (m)		Rotation Error (°)	
	Mean	Std.	Mean	Std.
Proposed (Ours)	0.121	0.077	0.30	0.144
Youngji Kim et al. [1]	0.15	0.14	0.34	0.40
D. Rozenberszki et al. [23]	~2.5	2.0	–	–
Xiaohu Lin et al. [45]	3.18	5.58	1.27	1.97

Note: Bold values indicate the best performance in each category.

leverages a prior map to maintain globally consistent localization without requiring revisit-based correction. As shown in Table 5.12, 5.13, and Figure 5.12, SLAM baselines such as KISS-SLAM and MOLA still exhibit several meters of drift, whereas our approach remains tightly aligned with the reference trajectory.

Table 5.12: Translation (APE) and Rotation Error statistics for **KITTI Sequence 05**

Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed	APE (m)	0.483	0.121	0.102	0.007	0.143	0.077
	Rot. (deg)	1.914	0.3	0.356	0.078	0.402	0.144
KISS-ICP	APE (m)	5.067	1.460	1.445	0.281	1.604	0.666
	Rot. (deg)	2.696	1.376	1.441	0.000	1.465	0.504
MOLA SLAM	APE (m)	6.817	1.539	1.447	0.306	1.793	0.920
	Rot. (deg)	3.861	1.931	1.886	0.000	2.015	0.577

Note: Bold values indicate the best performance across each metric.

Table 5.13: Translation (APE) Error statistics for **Mulran-KAiST-03**

Method	Metric	Max	Mean	Median	Min	RMSE	Std Dev
Proposed	APE (m)	0.74	0.16	0.12	0.03	0.19	0.1
MOLA	APE (m)	19.98	8.83	6.85	2.79	9.99	4.68
KISS	APE (m)	75.82	33.47	32.27	4.49	36.57	14.72

Note: Bold values indicate the best performance across each metric.

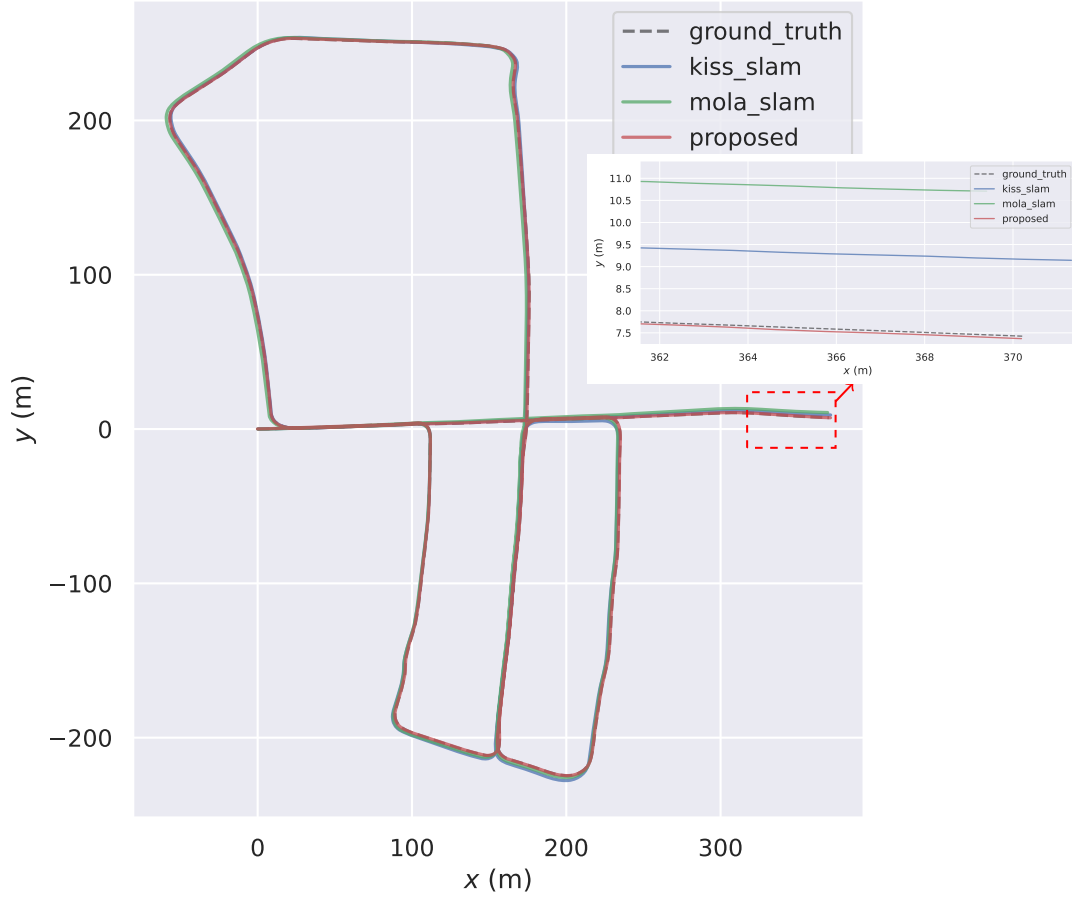


Figure 5.12: **KITTI Sequence 05 – Trajectory Alignment.** The figure shows the estimated trajectories from KISS-SLAM, MOLA SLAM, and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region..

5.3 Summary of Results

The overall performance of the proposed localization system across all evaluated datasets is summarized in Table 5.14. The system consistently achieves mean absolute pose errors between 5–16 cm and RMSE below 20 cm, demonstrating strong drift resilience and accuracy. Standard deviation values remain low across sequences, indicating stable and consistency.

Table 5.14: Summary of translation error metrics (APE) for the proposed method across all sequences.

Sequence	Mean APE (m)	RMSE (m)	Std. Dev (m)
Saxion Sequence 1	0.052	0.067	0.042
Saxion Sequence 2	0.090	0.107	0.058
Saxion Sequence 3	0.097	0.135	0.094
Saxion Sequence 4	0.04	0.05	0.033
KITTI Sequence 05	0.121	0.143	0.077
MulRan KAIST 03	0.160	0.190	0.100

Chapter 6

Discussion

This chapter reflects on the results presented in Chapter 4 and discusses how they address the research questions, the implications of the findings, and possible directions for future work.

6.1 Interpretation of Results

6.1.1 Accuracy and Robustness

The results demonstrate that integrating a prior 3D map into a LiDAR-inertial localization framework significantly enhances both accuracy and robustness. By aligning real-time LiDAR scans with globally consistent reference map, the system reduces drift that typically accumulates in odometry-only methods. Experimental results from multiple datasets show that the proposed system achieves centimeter- to decimeter-level translational RMSE (see Table 5.2, 5.3, 5.4), while standalone LiDAR-Inertial Odometry exhibits drift ranging from meter-level to over ten meters in extended trajectories. Unlike SLAM approaches that rely on loop closures, this method consistently aligns live LiDAR scans to a static reference map, maintaining accuracy even in environments with sparse or unreliable loop opportunities (Table 5.12).

To effectively combine high-frequency LiDAR-Inertial Odometry (FAST-LIO2) with NDT-based map matching, the proposed system employs a factor graph optimization framework. The factor graph fuses odometry and scan-matching constraints into a consistent probabilistic model, reducing accumulated linearization errors compared to filtering methods. The sliding-window strategy further ensures real-time performance by limiting the optimization to a fixed temporal window, maintaining constant computational cost regardless of trajectory length (see Figure 5.4a). This design effectively mitigates drift in odometry and

compensates for scan-matching failures, preserving both short-term precision and long-term consistency in challenging environments.

6.1.2 Real-Time Performance

Results show that while standard NDT and ICP struggle to meet real-time constraints as map size increases often resulting in slow convergence or failure, the proposed approach maintains reliable performance (Table 5.5). By operating on locally segmented submaps and leveraging multithreaded NDT-OMP, the system ensures fast and stable scan matching. To manage computational load in large-scale environments, the system incorporates dynamic submap loading and tile-based map management. Instead of operating on a full global map, only local map tiles within a defined radius are loaded and used for scan-to-map matching. This reduces memory usage and significantly improves convergence speed of multithreaded NDT-OMP matching. The proposed pipeline operates at an average rate of 48 Hz (23 ms per frame), as illustrated in Figure 5.4b.

6.1.3 Environmental Adaptability

Under moderate fog (visibility 60m), both NDT and the fusion method exhibit stable performance, with APE increasing only marginally compared to the baseline (Table 5.10). This confirms that NDT scan matching retains resilience in moderately degraded visibility. However, under severe fog (visibility 30m), scan matching becomes unreliable due to poor feature correspondences, and FAST-LIO2 suffers from high drift. As a result, the fusion pipeline repeatedly fails, unable to maintain reliable pose estimates.

In sparse map regions and transition zones, scan-to-map registration frequently fails due to a lack of correspondences. Nevertheless, the fusion method maintains localization thanks to high-rate odometry updates and prior structure in the factor graph (Figure 5.11).

Dynamic object removal further enhances registration performance. Filtering out transient objects improves convergence, reduces iteration counts, and lowers overall computational load during scan matching (see Table 5.7). While the object detection module introduces some processing overhead, it remains feasible for real-time operation, particularly on GPU-accelerated platforms. However, the system may face limitations on resource-constrained devices without hardware acceleration.

6.2 Implications

These findings underscore the value of combining prior maps, real-time odometry, and probabilistic optimization for autonomous navigation. The approach enables accurate and robust pose estimation in GNSS-denied and moderately degraded visual environments, which are common in urban outdoor and indoor settings. The use of dynamic submap loading and multithreaded NDT further supports the system’s deployment in real-time and large-scale environments, particularly for high-accuracy, repetitive navigation tasks.

6.3 Limitations and Future Work

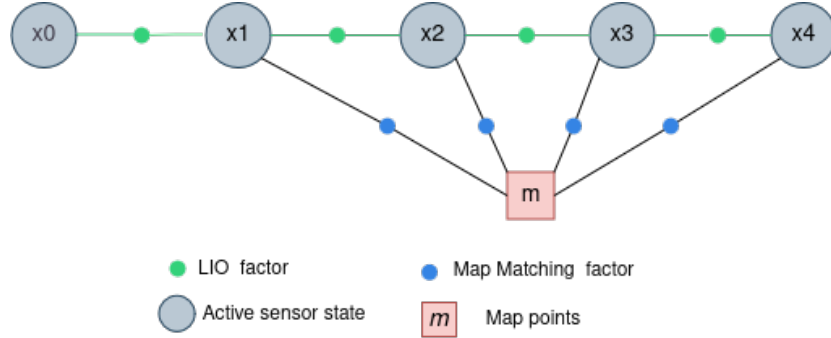
While the proposed localization system demonstrates high accuracy and robustness across diverse scenarios, several limitations remain. First, it assumes a known initial pose, which restricts its ability to perform global localization or recover from kidnapped robot scenarios. Second, the system relies on a static prior map without support for online map adaptation, which limits its effectiveness in long-term or evolving environments.

A further limitation arises in highly degenerate environments—such as corridors, open plains, or fog—where LiDAR lacks sufficient geometric richness. While FAST-LIO2 and multithreaded NDT are resilient under moderate degradation, their performance degrades in cases of extreme sparsity or planar dominance. FAST-LIO2 relies on consistent local structure for correction, and even NDT-OMP may fail to converge under heavy noise.

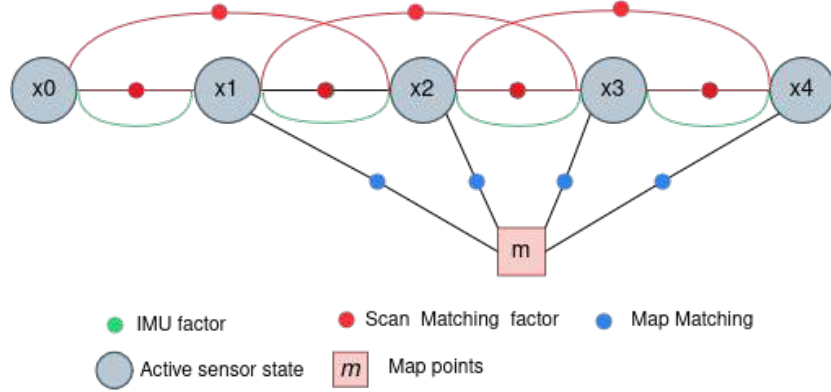
To address this, a future enhancement proposes replacing the tightly coupled LIO front-end with a modular graph-based framework, treating IMU data as a pre-integration factor. This would enable pose propagation during LiDAR degradation and support selective correction when reliable map constraints are available. The flexible structure also allows integration of visual or radar factors for improved localization in perceptually degraded scenes. The current and proposed formulations are shown in Figure 6.1, panels (a) and (b), respectively. In summary, future work may proceed along the following directions:

- **Global localization and recovery:** Integration of coarse initial pose estimation methods for startup or failure recovery.
- **Adaptive mapping:** Support for online map updates or hybrid SLAM fusion during long-term deployments.
- **Modular multi-sensor fusion:** A graph-based architecture incorporating visual, radar, and inertial factors to enhance robustness across sensor failures and environmental degradation.

These enhancements would strengthen the system’s applicability in real-world autonomous navigation, particularly in complex, dynamic, and partially observable environments.



(a) Current formulation: tightly coupled LiDAR-Inertial Odometry (FAST-LIO2) with direct map matching via NDT-OMP.



(b) Proposed formulation: modular factor graph where IMU is treated as a pre-integration factor, allowing flexible integration of additional sensors.

Figure 6.1: Comparison of sensor fusion architectures. (a) The current design integrates FAST-LIO2 with NDT-based map matching in a tightly coupled manner. (b) The proposed approach restructures the system into a modular factor graph, supporting IMU pre-integration and extensibility to visual or radar factors for increased robustness in degraded environments.

Chapter 7

Conclusion

7.1 Summary of Findings

This thesis presented a real-time, map-based localization system that combines LiDAR-Inertial Odometry (FAST-LIO2), multithreaded Normal Distributions Transform (NDT) scan matching, and a fixed-lag factor graph optimization framework. The system was designed to provide accurate and drift-resilient 6-DoF pose estimation in large-scale, GNSS-denied, and partially dynamic environments.

Evaluations conducted on benchmark and custom datasets demonstrated that the system consistently achieved centimeter- to decimeter-level translational RMSE, outperforming standalone odometry methods such as FAST-LIO2, which exhibited meter-level drift in long trajectories. The proposed method achieved similar or superior localization accuracy compared to recent map-based approaches. Furthermore, compared to SLAM-based techniques that rely on loop closures, the proposed method maintained better global consistency by continuously aligning LiDAR scans with a pre-built 3D map.

Real-time performance was sustained with an average processing latency of under 23 ms per frame, enabled by a multithreaded NDT implementation and dynamic submap loading. This ensured scalability across large environments while maintaining memory and computational efficiency. Additionally, dynamic object removal using a deep learning-based detection model improved registration stability in semi-dynamic scenes, and NDT demonstrated resilience under moderate visibility degradation such as fog. The system also maintained robust localization in feature-sparse map boundaries and unmapped transition zones, where scan-to-map registration typically fails. By relying on high-frequency odometry within the factor graph, it preserved pose continuity and minimized drift.

These findings confirm that the proposed localization pipeline is well-suited for high-accuracy, real-time navigation in complex, unstructured, and dynamic environments.

Appendix A

Trajectory Plot

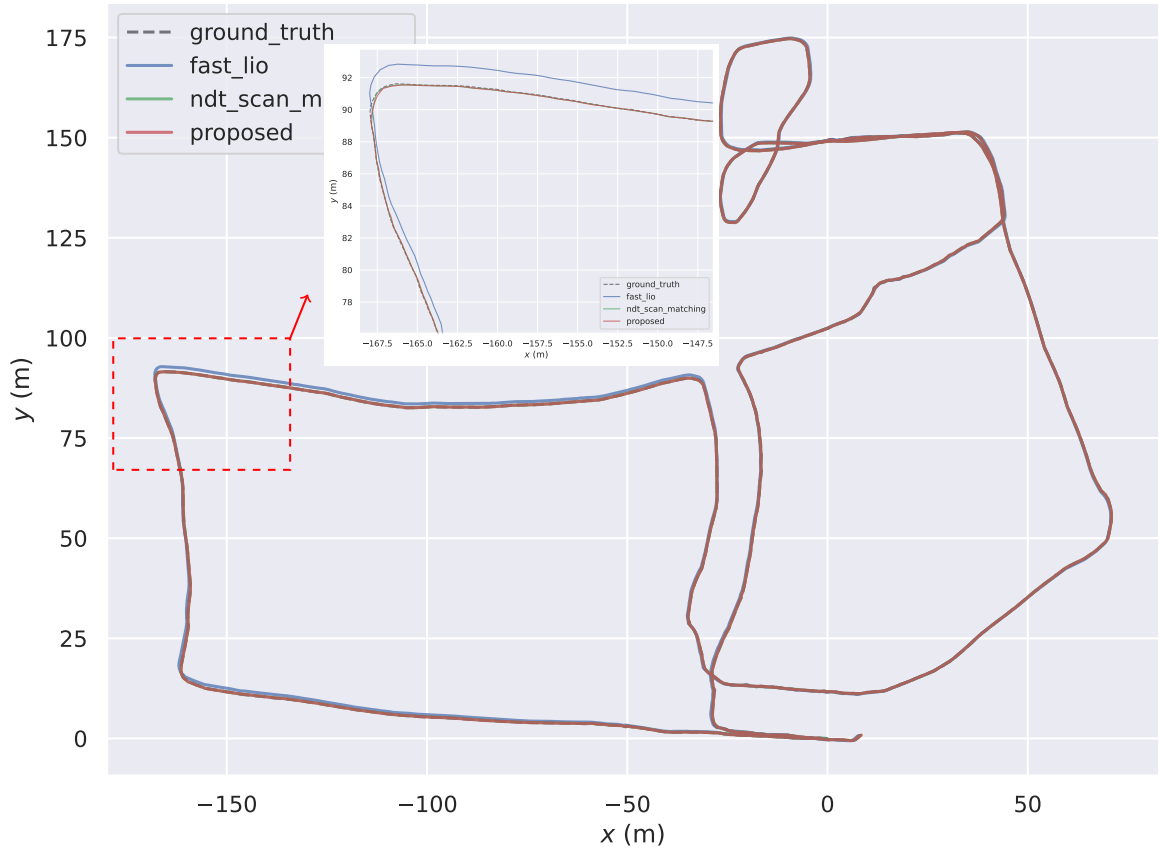


Figure A.1: **Saxion Sequence 02 – Trajectory Alignment.** The figure shows the estimated trajectories from FAST-LIO2, low frequency NDT map matching , and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region.

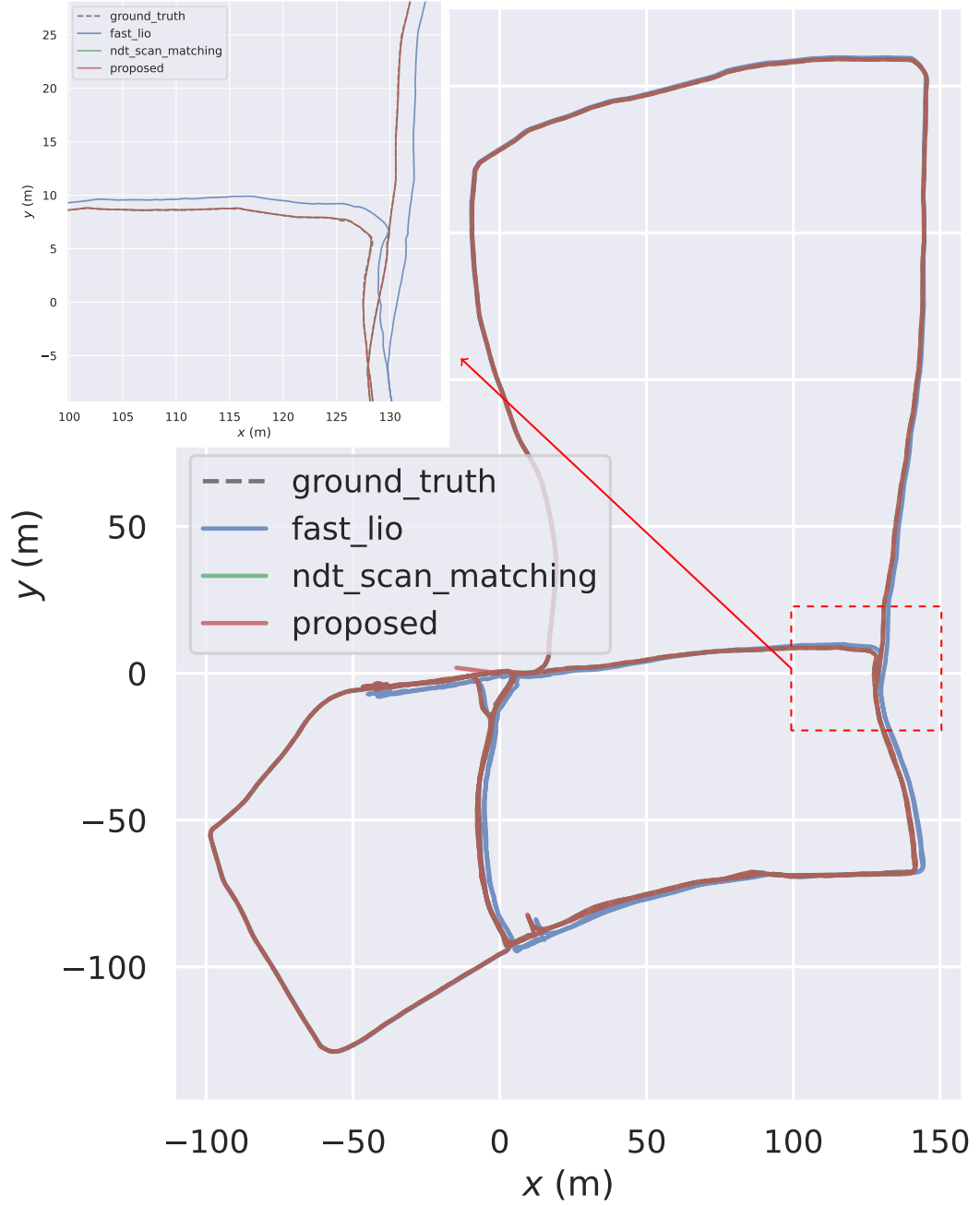


Figure A.2: **Saxion Sequence 03 – Trajectory Alignment.** The figure shows the estimated trajectories from FAST-LIO2, low frequency NDT map matching , and the proposed method, overlaid against ground truth. The red dashed box indicates the zoomed region.

Appendix B: Declaration Form

The signed declaration form is attached as a separate document titled `Eliyas-TDF.pdf`.

Bibliography

- [1] Y. Kim, J. Jeong, and A. Kim. “Stereo camera localization in 3D LiDAR maps”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, Oct. 2018, pp. 1–9.
- [2] H. Carvalho et al. “Optimal nonlinear filtering in GPS/INS integration”. In: *IEEE Transactions on Aerospace and Electronic Systems* 33 (1997), pp. 835–850. DOI: [10.1109/7.599911](#).
- [3] H. Liu et al. “A Precise and Robust Segmentation-Based Lidar Localization System for Automated Urban Driving”. In: *Remote Sensing* 11.12 (2019), p. 1348. DOI: [10.3390/rs11111348](#).
- [4] J. Levinson, M. Montemerlo, and S. Thrun. “Map-based precision vehicle localization in urban environments”. In: *Proceedings of Robotics: Science and Systems*. Vol. 4. Cambridge, MA, USA, 2007, pp. 1–8.
- [5] A. Mohamed et al. “A Review on Visual and Non-visual Odometry Methods”. In: *Journal of Robotic Systems* (2020).
- [6] X. Wang et al. “Comprehensive Survey of Visual Odometry: Trends and Evaluation”. In: *IEEE Access* (2019).
- [7] Paul J. Besl and Neil D. McKay. “A Method for Registration of 3-D Shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1992.
- [8] Ji Zhang and Sanjiv Singh. “LOAM: Lidar Odometry and Mapping in Real-time”. In: *Robotics: Science and Systems*. 2014.
- [9] J. Tang et al. “A Loosely Coupled LiDAR-Inertial Fusion Method”. In: *Sensors* (2015).
- [10] T. Shan et al. “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2020).

- [11] Tong Qin, Peiliang Li, and Shaojie Shen. “LINS: A LiDAR-Inertial State Estimator for Robust and Fast Localization”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3867–3874.
- [12] Wei Xu et al. “FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324.
- [13] Wei Xu et al. “FAST-LIO2: Generalized Lightweight LIO system on ROS2”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2022, pp. 5958–5964.
- [14] Cesar Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332. DOI: [10.1109/TR0.2016.2624754](https://doi.org/10.1109/TR0.2016.2624754).
- [15] Wolfgang Hess, Stefan Kohlbrecher, et al. “Real-Time Loop Closure in 2D LIDAR SLAM”. In: *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278.
- [16] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters”. In: *IEEE Transactions on Robotics*. 2007, pp. 34–46.
- [17] Jose Luis Blanco-Claraco. “A flexible framework for accurate LiDAR odometry, map manipulation, and localization”. In: *The International Journal of Robotics Research* 0.0 (2025), p. 02783649251316881. DOI: [10.1177/02783649251316881](https://doi.org/10.1177/02783649251316881). eprint: <https://doi.org/10.1177/02783649251316881>. URL: <https://doi.org/10.1177/02783649251316881>.
- [18] T. Guadagnino et al. “KISS-SLAM: A Simple, Robust, and Accurate 3D LiDAR SLAM System With Enhanced Generalization Capabilities”. In: *arXiv preprint arXiv:2503.12660* (2025). URL: <https://arxiv.org/pdf/2503.12660>.
- [19] Kenji Koide et al. “Tightly Coupled Range Inertial Localization on a 3D Prior Map Based on Sliding Window Factor Graph Optimization”. In: *arXiv preprint arXiv:2402.05540* (2024).
- [20] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
- [21] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. “Map-Based Precision Vehicle Localization in Urban Environments”. In: *Proceedings of the Robotics: Science and Systems (RSS)*. 2007.

- [22] Xiaohu Lin et al. “Autonomous Vehicle Localization with Prior Visual Point Cloud Map Constraints in GNSS-Challenged Environments”. In: *Remote Sensing* 13.3 (2021), p. 506. DOI: [10.3390/rs13030506](https://doi.org/10.3390/rs13030506).
- [23] Dávid Rozenberszki and András L. Majdik. “LOL: Lidar-only Odometry and Localization in 3D Point Cloud Maps”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2020.
- [24] Peter Biber and Wolfgang Straßer. “The normal distributions transform: A new approach to laser scan matching”. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. Vol. 3. IEEE. 2003, pp. 2743–2748.
- [25] Martin Magnusson et al. “Scan registration for autonomous mining vehicles using 3D-NDT”. In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827.
- [26] Kenji Koide, Jun Miura, and Emanuele Menegatti. “A Portable 3D LIDAR-based System for Long-term and Wide-area People Behavior Measurement”. In: *Advanced Robotic Systems* (2019).
- [27] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. “Center-based 3D Object Detection and Tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11784–11793.
- [28] Alex H Lang et al. “PointPillars: Fast Encoders for Object Detection from Point Clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 12697–12705.
- [29] Yan Yan, Yuxing Mao, and Bo Li. “SECOND: Sparsely Embedded Convolutional Detection”. In: *Sensors* 18.10 (2018), p. 3337.
- [30] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11621–11631.
- [31] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. DOI: [10.1115/1.3662552](https://doi.org/10.1115/1.3662552).
- [32] Anastasios I. Mourikis and Stergios I. Roumeliotis. “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2007, pp. 3565–3572. DOI: [10.1109/ROBOT.2007.364024](https://doi.org/10.1109/ROBOT.2007.364024).

- [33] Simon J. Julier and Jeffrey K. Uhlmann. “A New Extension of the Kalman Filter to Nonlinear Systems”. In: *Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*. 1997, pp. 182–193.
- [34] Frank Dellaert. *Factor Graphs and GTSAM: A Hands-On Introduction*. <https://doi.org/10.5281/zenodo.14117>. Accessed: 2024-04-01. 2017.
- [35] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. “Factor Graphs and the Sum-Product Algorithm”. In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519. DOI: [10.1109/18.910572](https://doi.org/10.1109/18.910572).
- [36] Michael Kaess et al. “iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree”. In: *The International Journal of Robotics Research* 31.2 (2012), pp. 217–236. DOI: [10.1177/0278364911430419](https://doi.org/10.1177/0278364911430419).
- [37] Simon Herbert, Maurizio Scalea, and Cyrill Stachniss. “Fusion of stereo vision and lidar in large-scale urban environments”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2013, pp. 2559–2565.
- [38] MMDetection3D Contributors. *MMDetection3D: OpenMMLab next-generation platform for general 3D object detection*. <https://github.com/open-mmlab/mmdetection3d>. 2020.
- [39] Autoware Foundation. *Autoware Universe: An Open-source Software Stack for Autonomous Driving*. <https://github.com/autowarefoundation/autoware-universe>. Accessed: 2024-04-29. 2023.
- [40] Johannes Teufel, André Meißner, and Frank Wagner. “Realistic Rain, Snow, and Fog Variations for Comprehensive Performance Characterization of LiDAR Perception”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 567–574. DOI: [10.1109/IV53394.2022.00086](https://doi.org/10.1109/IV53394.2022.00086).
- [41] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.
- [42] Giseop Kim et al. “MulRan: Multimodal Range Dataset for Urban Place Recognition”. In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1394–1407. DOI: [10.1177/02783649211012805](https://doi.org/10.1177/02783649211012805).
- [43] Xinyu Liu et al. “Large-Scale LiDAR Consistent Mapping Using Hierarchical LiDAR Bundle Adjustment”. In: *IEEE Robotics and Automation Letters* 8.3 (2023), pp. 1523–1530. DOI: [10.1109/LRA.2023.3238902](https://doi.org/10.1109/LRA.2023.3238902).

- [44] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM*. <https://github.com/MichaelGrupp/evo>. 2017.
- [45] X. Lin et al. “Autonomous Vehicle Localization with Prior Visual Point Cloud Map Constraints in GNSS-Challenged Environments”. In: *Remote Sensing* 13.506 (2021). DOI: [10.3390/rs13030506](https://doi.org/10.3390/rs13030506).

4

**Indirect Visual-Inertial Simultaneous Localization and Mapping for
Dense 3D Reconstruction**
Goitom Abrha Leaku



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTERS DEGREE IN
INTELLIGENT FIELD ROBOTIC SYSTEMS

Indirect Visual-Inertial Simultaneous Localization and Mapping for Dense 3D Reconstruction

Author:

Goitom Abrha Leaku

Masters in Intelligent Field Robotic Systems

Internal supervisor:

Hajder Levente

Associate Professor

External supervisor:

Mohammed Aldibaja

Associate Professor at SMART Saxion

Budapest, 2025

Abstract

In the field of robotics, accurate localization and mapping are essential, particularly by using inexpensive, lightweight sensors such as cameras. This thesis work presents an indirect visual inertial simultaneous localization and mapping framework designed for dense 3D reconstruction across diverse environments. Unlike direct methods that rely heavily on photometric consistency, our approach leverages feature-based tracking and robust inertial data fusion to enhance system stability and scalability in diverse lighting and motion conditions. By combining feature-based visual tracking with inertial data in a tightly coupled optimization framework, the system achieves robust and accurate pose estimation under varying lighting and motion conditions. Dense 3D maps are generated by volumetric fusion of camera-derived depth information. The depth data from each frame of the RGB-D camera are converted into point clouds, which are then used to construct a dense map using iterative closest point (ICP) registration. The method was evaluated using custom datasets collected by mobile robots with ROS2 bag files in indoor and outdoor environments, and the TUM RGB-D benchmark data set with ground truth for quantitative precision analysis. The results demonstrate superior localization accuracy and 3D reconstruction compared to state-of-the-art methods, validating the robustness and versatility of the framework.

Keywords:

Visual Simultaneous Localization and Mapping, Inertial Measurement Unit, Sensor fusion, Iterative closest point, Dense mapping

Acknowledgment

First and foremost, I thank God for his guidance and blessings throughout my journey. I owe a special thanks to my family for their unwavering love, patience, and steadfast support throughout my life and academic pursuits.

I would like to express my sincere gratitude to my supervisors, Associate Professor Mohammad Aldibaja from the Saxion Smart Mechatronics and Robotics research group and Associate Professor Hajder Levente from the Faculty of Informatics at Eötvös Loránd University, for their invaluable guidance, continuous support, and encouragement throughout this research. Their insights and feedback were instrumental in shaping the direction of this work and greatly enriched my research experience.

I also extend my heartfelt thanks to Dr. Narcís Palomeras and Imma Güell for their Constant support and encouraging words throughout my studies: this master's journey would not have been possible without them. I am equally grateful to Dr. Zoltán Istenes and Reka from Eötvös Loránd University for making my time in Budapest an unforgettable experience. Special appreciation goes to Professor Abeje Y. Mersha and Dr. Gerdine Meijer from the Saxion Smart Mechatronics and Robotics Research Group for their guidance and for fostering a supportive and enriching research environment.

I thank the members of the Saxion Smart Mechatronics and Robotics group for their collaboration, constructive discussions, and invaluable technical assistance. Special thanks go to my friends and colleagues for their moral support and for fostering a positive and motivating working environment.

Finally, I am deeply thankful to the Erasmus Mundus Joint Master's program, particularly the Intelligent Field Robotic Systems, for their invaluable support throughout my Master's journey. I would also like to express my sincere appreciation to the Saxion Smart Mechatronics and Robotics group for providing the resources, guidance, and funding that made this thesis work and research possible.

Contents

Abstract	i
Acknowledgment	ii
Acronyms	iii
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	4
1.3 Research Questions	4
1.4 Contributions	5
1.5 Thesis Outline	5
2 Literature Review and Background	7
2.1 Literature Review	7
2.1.1 Visual Odometry	7
2.1.2 Visual Simultaneous Localization and Mapping (VSLAM)	8
2.1.3 Registration	10
2.2 Background	11
2.2.1 Sensing Modalities in VSLAM	11
2.2.2 VSLAM and Visual-Inertial SLAM	14
3 Methodology	18
3.1 System Setup	18
3.1.1 Software Setup	18
3.1.2 Data Collection and Pre-processing	19
3.1.3 Hardware Setup	19
3.2 Main Pipeline Features Based (Indirect) VSLAM	21
3.2.1 Feature Detection	23

3.2.2	Feature Matching and Tracking	23
3.2.3	Random Sample Consensus (RANSAC)	25
3.2.4	Pose Estimation	25
3.2.5	Inertial Measurement Unit (IMU) Pre-integration	26
3.2.6	Visual Inertial Fusion	27
3.2.7	Mapping	28
3.2.8	Local and Global Mapping	29
3.2.9	Bag of Words	32
3.2.10	Pose Graph Optimization and Bundle Adjustment	33
4	Results and Discussions	35
4.1	Mapping and Localization with Custom Data Collected	35
4.1.1	Indoor Experiment	35
4.1.2	Outdoor Experiment	41
4.2	Benchmark Dataset-Based Evaluation	44
4.3	Limitations	53
5	Conclusion and Future Work	54
5.1	Conclusion	54
5.2	Future Work	56
	Bibliography	57

Acronyms

APE Absolute Pose Error.

AR augmented reality.

BA Bundle Adjustment.

BOW Bag of Words.

DBoW Distributed Bag of Words.

DOF Degree of Freedom.

EKF Extended Kalman Filter.

EM Essential Matrix.

FM Fundamental Matrix.

ICP Iterative Closest Point.

IMU Inertial Measurement Unit.

KLT Kanade-Lucas-Tomasi Tracker.

LiDAR Light Detection and Ranging.

ORB Oriented FAST and Rotated BRIEF.

PCL Point Cloud Library.

PGO Pose graph Optimization.

RANSAC Random Sample Consensus.

RGB Red Green Blue.

RGB-D Red Green Blue and Depth.

RMSE Root Mean Square Error.

ROS 2 Robot Operating System 2.

ROSBag ROS Data Recording and Playback Tool.

RViz 2 ROS Visualization Tool 2.

SIFT Scale-Invariant Feature Transform.

SLAM Simultaneous Localization and Mapping.

SURF Speeded-Up Robust Features.

UAV Unmanned Aerial Vehicle.

VINS Visual Inertial Navigation.

VO Visual Odometry.

VSLAM Visual Simultaneous Localization and Mapping.

XYZ Cartesian coordinate axes (X, Y, Z).

List of Figures

2.1	Monocular camera	12
2.2	Stereo-ZED2 camera	13
2.3	Inertial Measurement Unit	14
2.4	Visual-Inertial fusion [36]	15
2.5	Visual comparison of re-projection error representations.	16
2.6	Direct visual SLAM principle [45]	17
3.1	Main Hardware Setup. Figure 3.1a shows the ground robot equipped with the drone used to collect Robot Operating System 2 (ROS 2) bags. Figure 3.1b shows the main sensors integrated with the mobile ground robot.	19
3.2	ZED Camera[47], sensor used for data collection.	20
3.3	General VSLAM diagram for both the Front-End and Back-End components	22
3.4	Proposed method Pipeline: (2) -> represents when there is a new frame from the input camera sensor, the frame is processed to give the dense local map which uses source to Iterative Closest Point (ICP), where as (3) -> uses as the previous frames aligned dense map are used as a target to the ICP registration. The (4) and (5) -> are a collection of Distributed Bag of Words (DBoW) vectors for each keyframes which are used to compare with the current keyframe for loop closure detection, and for optimization until the current keyframe respectively. Most methods (white boxes) are adapted from [18].	22
3.5	ORB feature detection in different environments: outdoor (3.5a) and indoor (3.5b) scenes.	23
3.6	Oriented FAST and Rotated BRIEF (ORB) feature detection and matching pipeline.	25
3.7	Feature Matching [49].	25
3.8	Local mapping [55]	30
3.9	Red Green Blue and Depth (RGB-D) camera based local mapping pipeline.	30

3.10	ICP Based Point cloud registration [56]	32
3.11	Pose graph [61]	33
4.1	Comparison between the frame's grayscale image and the reconstructed local mapping: the green line indicates the camera path, and the resulting map is the frame's local mapping.	36
4.2	ICP alignment using consecutive frames captured with an RGB-D camera over the full field of view.	37
4.3	Dense map of small trajectories in an indoor environment using ICP, reconstructed with a limited camera field of view.	38
4.4	Reconstructed map showing increased trajectory length and extended vertical field of view, as a continuation of Figure 4.3.	38
4.5	Visualization for qualitative comparison between the reconstructed dense map and the corresponding raw image. The view is zoomed in to highlight details: <i>left — accumulated dense map; right — raw image of the environment.</i>	39
4.6	Reconstructed dense map of the indoor scene from a side view for an indoor experiment, generated using the method proposed in Diagram 3.4(green). .	40
4.7	Zoomed-in side view of the indoor scene from Figure 4.6, highlighting structural details for improved visualization.	40
4.8	Comparison of the predefined GPS path and the reconstructed dense map with estimated trajectory.	41
4.9	Zoomed-in view of the reconstructed dense map for improved visualization	42
4.10	Estimated 3D trajectory of the robot in the outdoor environment.	43
4.11	Estimated trajectories along the Cartesian coordinate axes (X, Y, Z) (XYZ), showing pose evolution with respect to each reference axis.	43
4.12	Robot speed over time for the entire trajectory, illustrating velocity variations at each time step throughout the outdoor movement experiment . . .	44
4.13	RGB image based environment and corresponding reconstructed dense map for <i>freiburg2_xyz</i> dataset	45
4.14	3D trajectory and ground truth for the <i>freiburg2_xyz</i> sequence.	46
4.15	Absolute Trajectory Error (ATE) per frame on the <i>fr2_xyz</i> sequence. . . .	46
4.16	XYZ position components of the <i>freiburg2_xyz</i> sequence along with ground truth, used to evaluate the accuracy of the estimated trajectory at each time step.	47

4.17	Per-axis XYZ error over time for the <i>freiburg2_xyz</i> sequence.	47
4.18	RGB images from different viewpoints in the <i>freiburg3_office_household</i> sequence from the TUM RGB-D dataset.	48
4.19	Reconstructed dense maps of the <i>freiburg3_office_household</i> sequence from different viewpoints from TUM RGB-D datasets.	49
4.20	3D (XYZ) estimated trajectory and ground truth for <i>Fr3_household</i>	49
4.21	Per-frame error for <i>Fr3_household</i>	50
4.22	XYZ position components of the <i>freiburg3_office_household</i> sequence along with ground truth, used to evaluate the accuracy of the estimated trajectory at each time step.	50
4.23	Per-axis XYZ error over time for the <i>freiburg3_long_house</i> sequence. . . .	51

List of Tables

3.1	Summary of datasets used for evaluation	19
3.2	Basic Specifications of the ZED 2 Camera [47]	20
4.1	Root Mean Square Error (RMSE) comparison (in meters) on the <i>freiburg2_xyz</i> dataset using different dense VSLAM methods, the RMSE of iMAP [64], NICE-SLAM [65], and ESLAM [66] are adapted [66]. Bold achieves the lowest RMSE(higher accuracy).	48
4.2	RMSE comparison (in meters) on the <i>freiburg3_office_household</i> dataset using different dense VSLAM methods. The RMSE values are adapted from [66]. The proposed method (bold) achieves the lowest RMSE, indicating superior performance.	51
4.3	Absolute Pose Error (APE) and RMSE for TUM RGB-D benchmark sequences, with translational (m) and rotational (deg) values. <i>trans</i> —> <i>Translational</i> and <i>rot</i> —> <i>Rotational</i>	52
4.4	RMSE comparison of our method with state-of-the-art approaches on selected TUM RGB-D sequences. Lower RMSE values indicate better trajectory accuracy. Missing entries (denoted by '-') indicate that the corresponding method was not evaluated on that sequence.	52

Chapter 1

Introduction

Robotics is an interdisciplinary field that combines principles from engineering, computer science, and artificial intelligence to design, construct, and operate robots. A robot is a programmable machine capable of executing a series of actions either autonomously or with minimal human intervention. In recent years, the use of robots has expanded significantly across various domains, including industrial automation, healthcare, agriculture, space exploration, military operations, and entertainment. The primary driving force behind the advancement of robotics is the need to improve efficiency, accuracy, and safety in tasks that are often dangerous to humans, monotonous, or demand a high degree of precision.

1.1 Motivation

The growing complexity of modern tasks and the demand for automation in both industrial and domestic environments have heightened interest in robotics. Robots can operate in environments in deep-sea locations, disaster zones, or space—thereby reducing risk and increasing productivity. Additionally, as robotics technology becomes more accessible and cost-effective, its integration into everyday life and work processes continues to grow.

Field Robotics, as a subdomain of mobile robotics, is focused on developing autonomous robots capable of performing complex tasks in diverse and often challenging environments. These robots must not only navigate efficiently but also adapt to a wide range of dynamic and unpredictable conditions, making the development of reliable and robust SLAM algorithms a key research priority. The ability to navigate through unknown

environments, construct a map of the surroundings, and estimate one's own position and orientation within that map is a fundamental capability observed in many intelligent biological species. This ability is equally critical in the development of autonomous robotic systems, particularly in the domain of mobile robotics. A robot must be able to operate reliably in environments where pre-existing maps or external localization systems (such as GPS) are unavailable or unreliable. This challenge—enabling a robot to concurrently map an environment while determining its own pose within it—is known as the Simultaneous Localization and Mapping (SLAM) problem. Solving SLAM involves fusing data from onboard sensors, such as cameras, Light Detection and Ranging (LiDAR), or IMU), to estimate both the robot's trajectory and the structure of the surrounding environment. Visual SLAM, in particular, leverages cameras as the primary sensing modality, offering a cost-effective and information-rich alternative to traditional sensors. SLAM is a foundational component for enabling true autonomy in a wide range of applications, from autonomous vehicles and drones to planetary rovers and household service robots.

The significance of SLAM is reflected in its widespread adoption across both research and industrial domains. It plays a critical role in enabling autonomy for a diverse range of platforms, such as planetary rovers used in space exploration, autonomous vacuum cleaners in domestic settings, unmanned aerial vehicles (UAVs) for surveillance, and underwater robots for reef monitoring and exploration in hazardous or inaccessible environments. A particularly fascinating area of interest within SLAM research is the use of cost-effective sensors. VSLAM systems leverage visual data to perform localization and mapping, offering a practical and economical alternative to more expensive sensor suites such as LiDAR.

Let's consider a robot tasked with navigating an unfamiliar place to do some predefined task. To complete its task, the robot must accurately determine its position within the environment. As other methods like GPS could assist in localization, they may be unreliable or provide insufficient accuracy for the given task. To accurately localize the robot within the environment can be achieved through the integration of onboard sensors, enabling the robot to process data from its environment and maintain accurate localization. While odometry, which is localizing the robot with incremental motion, can be achieved. This enables the robot to track its movement, but over time, errors accumulate, causing the estimated trajectory to diverge from the robot's actual path. Odometry is effective for short-term motion estimation, but for long-term operation in a consistent environment, a map is needed to ensure accuracy and correct the accumulated drift.

In addition, mapping is the most important in autonomous navigation, perception, and

decision-making. In field robotics, Various types of maps are utilized depending on the specific requirements of the application, available sensors, and environmental constraints. *Visual maps* are constructed using data captured from cameras, including monocular, stereo, or RGB-D sensors. These maps form the foundation of VSLAM, where the robot uses visual features—such as corners, edges, or textures—to estimate its trajectory and reconstruct the surrounding environment. Visual mapping is especially interesting due to the affordability and lightweight nature of cameras, making it ideal for drones and mobile robots. However, visual SLAM systems face challenges such as sensitivity to lighting changes, dynamic scenes, and scale ambiguity in monocular setups, which may affect accuracy and robustness. *LiDAR-based maps*, on the other hand, are created using laser range sensors, which measure the distance to objects by analyzing the return time of laser pulses. LiDAR systems can produce highly accurate 2D or 3D representations of the environment, making them particularly effective in outdoor navigation, autonomous vehicles, and robotics applications that require precise spatial information. LiDAR maps are generally more robust to lighting conditions and can operate reliably in both day and night. Despite their advantages, LiDAR sensors are often more expensive and computationally demanding compared to visual systems.

LiDAR-based maps, on the other hand, are created using laser range sensors, which measure the distance to objects by analyzing the return time of laser pulses. LiDAR systems can produce highly accurate 2D or 3D representations of the environment, making them particularly effective in outdoor navigation, autonomous vehicles, and robotics applications that require precise spatial information. LiDAR maps are generally more robust to lighting conditions and can operate reliably in both day and night. Despite their advantages, LiDAR sensors are often more expensive and computationally demanding compared to visual systems.

occupancy grid maps represent the environment as a grid where each cell contains a probability value indicating whether it is occupied, free, or unknown. This probabilistic framework is widely used in SLAM and path planning due to its simplicity and effectiveness in 2D environments. However, occupancy grids can be memory-intensive and may not scale efficiently to large or 3D environments without optimization.

Semantic maps go beyond geometric information by incorporating labels and contextual understanding of objects within the environment, such as identifying doors, chairs, tables, or rooms. These maps are particularly useful for service robots operating in human-centric environments, where understanding the function and identity of objects can en-

hance task performance and interaction. Building semantic maps typically involves integrating machine learning or deep learning-based perception systems to detect and classify objects in real time.

1.2 Problem Statement

Despite significant advancements in VSLAM, several challenges persist—particularly in environments with poor texture, rapid motion, or large featureless regions. These conditions degrade localization accuracy and make it difficult to generate reliable maps. Moreover, many applications, such as autonomous navigation, path planning, and augmented reality, require dense 3D maps rather than sparse or semi-dense ones.

Direct methods like LSD-SLAM [1] provide semi-dense reconstruction but often lack robustness in large-scale or dynamic environments. On the other hand, feature-based approaches such as ORB-SLAM and ORB-SLAM2 [2, 3] achieve greater scalability and robustness by leveraging keypoints, yet they struggle to produce dense reconstructions.

Integrating an IMU with visual data has improved SLAM system robustness, especially under rapid motion. However, it also introduces additional complexity in sensor fusion, including accurate synchronization, initialization, and error propagation. IMU-only systems are also highly susceptible to drift and noise, making visual-inertial fusion essential for accurate pose estimation.

The main objectives of this thesis are as follows:

- To design and implement a visual-inertial SLAM system that integrates multiple visual sensors to achieve reliable localization and dense 3D reconstruction.
- To develop and optimize sensor fusion algorithms—primarily between camera and IMU data—to enhance pose estimation accuracy and reduce localization errors, particularly in lateral and longitudinal directions.
- To test and evaluate the proposed SLAM system on a real robotic platform across various environmental scenarios.

1.3 Research Questions

This research aims to investigate how VSLAM performance can be improved by integrating methods for featureless areas and refining orientation angle estimation using

Visual-Inertial Odometry (VIO). The key research questions addressed in this thesis are

- How can the integration of visual sensory data and IMU measurements improve the accuracy and robustness of SLAM in different environments, including featureless environments?
- How can loop closure detection improve map consistency in SLAM systems?

1.4 Contributions

This thesis makes the following key contributions to the field of visual-inertial SLAM and dense 3D reconstruction:

- The thesis introduces a frame- and keyframe-based strategy for dense map generation that evaluates the computational cost with reconstruction for each of these methods. This approach allows for scalable 3D mapping while maintaining high spatial detail by taking each frame and also selectively processing keyframes with optimized depth fusion techniques.
- Optimized and containerized existing SLAM systems with enhanced visual-inertial integration, improving localization accuracy in low-texture and dynamic environments.
- Comprehensive experiments are conducted on the TUM RGB-D benchmark to validate the proposed system's performance. Quantitative metrics such as Absolute Pose Error (APE) for both translational and rotational errors are used to assess accuracy, and comparisons with state-of-the-art methods demonstrate improved performance in both pose estimation and map quality.

1.5 Thesis Outline

This thesis is organized into the following chapters:

- **Chapter 2 – Literature Review and Background:** This chapter presents a comprehensive overview of existing visual odometry and VSLAM techniques. It also discusses point cloud registration methods and the fundamental concepts of VSLAM systems and sensor used.

- **Chapter 3 – Methodology:** This chapter describes the proposed approach in detail, covering the hardware setup, software architecture, algorithmic components, and implementation aspects of the developed pipeline.
- **Chapter 4 – Results and Discussion:** The experimental results are presented and analyzed in this chapter, including evaluations on both custom-collected datasets and publicly available benchmark datasets. It also discusses the observed limitations of the proposed method.
- **Chapter 5 – Conclusion and Future Work:** This chapter summarizes the key contributions and findings of the thesis. It also outlines potential directions for future research and improvements.

Chapter 2

Literature Review and Background

This chapter reviews key research in visual odometry, visual and visual-inertial SLAM, and registration, while also presenting the theoretical foundations essential for understanding VSLAM techniques.

2.1 Literature Review

2.1.1 Visual Odometry

Visual odometry is the process of estimating the position of an agent (which includes a vehicle, human, and robot) using only the input of a single or multiple cameras [4] without considering loop closure when the agent or robot revisits the previous position. Many visual odometry methods have been proposed, which can be divided into monocular [5] and stereo camera methods [6]. These methods are then further divided into feature matching (matching features over a number of frames) [7], feature tracking [8] (tracking features in a new frame), and optical flow techniques, which are intensity-based for all pixels or specific regions in sequential images. The initial stage of visual odometry is feature detection; these features could be Harris corners [9] or ORB features [10] in each frame. This type of point feature has been found to provide relatively stable detections under small to moderate image distortions. Visual Odometry (VO) provides an incremental online estimate of the position of aa (robot) by analyzing the image sequences captured by a camera [10].

2.1.2 VSLAM

VSLAM uses cameras to estimate motion and build maps. Recent advances have been driven by increased computing power and the growing need for autonomous navigation in unknown environments across various application areas, including aerial, underwater, and terrestrial domains.

Accuracy can be significantly enhanced, especially in dynamic and fast-moving environments, by fusing camera data with an IMU, a method known as visual-inertial SLAM [11]. In [12], the feasibility of real-time VSLAM using a single camera was demonstrated. This work introduced a framework based on the Extended Kalman Filter (EKF) to simultaneously estimate the camera's pose and a sparse map of feature points. The approach utilized Shi-Tomasi feature detection to track reliable visual features over time, enabling robust monocular SLAM. The subsequent development of MonoSLAM [13] extended this framework by incorporating improvements in feature tracking, motion modeling, and real-time processing.

Keyframe-based methods build the map using a limited number of selected frames, while discarding data from intermediate frames. This strategy enables the application of more computationally intensive, yet highly accurate, Bundle Adjustment (BA) optimization at the keyframe rate.

In [2] and [3], ORB features are used, whose descriptor provides short-term and mid-term data association, builds a co-visibility graph to limit the complexity of tracking and mapping, and performs loop closing and relocalization using the bag-of-words library DBoW2 [14], achieving long-term data association. Visual SLAM system integrating the three types of data association for better accuracy.

Direct methods rely on image pixel intensities, minimizing photometric error rather than extracting features. [1] demonstrated large-scale semi-dense mapping using high-gradient pixels, though its reliance on pose graph optimization limited accuracy compared to [3]. Semi-direct visual odometry ([15], [16]) combines direct tracking of gradient pixels with FAST feature extraction, optimizing both camera trajectory and 3D structure via re-projection error.

In [17], a visual-inertial SLAM system able to reuse a map with short-term, mid-term, and long-term data association, using them in an accurate local visual-inertial BA based on IMU pre-integration, was proposed. Compared to the direct method [1], the indirect method, as in [18], is less robust in low-texture environments. However, direct methods,

which rely on pixel intensity, are limited to low- and medium-term data association. Additionally, compared to [19] and [20], the IMU initialization technique in [17] is too slow, which negatively affects robustness and accuracy. Faster initialization techniques were proposed in [19] and [20], based on a closed-form solution to jointly estimate scale, gravity, accelerometer bias, initial velocity, and visual feature depth. A very accurate and robust monocular-inertial odometry system with loop closing using DBoW and Pose graph Optimization (PGO) for four Degree of Freedom (DOF) and map-merging was proposed in Visual Inertial Navigation (VINS)-Mono[21], with Lucas-Kanade tracker, slightly more robust than descriptor matching. In VINS-Fusion [22].

For initial value estimation, a linear estimator initialization method that utilizes relative rotations from short-term IMU pre-integration was proposed in [23] and [24]. In [21] and [25], loop closure constraints are incorporated in both the local sliding window and global batch optimization. During the local optimization, feature observations from keyframes provide implicit loop-closure constraints, while keeping the problem size manageable by assuming the keyframe poses are perfect, thus excluding them from optimization. Inertial pre-integration, a computationally effective substitute for conventional integration of inertial measurement, was initially created by Lupton and Sukkarieh [26]. There is no need to reintegrate the state dynamics at every optimization stage because this method discretely integrates the inertial measurement dynamics in a local frame of reference. Although it solves the problem of computing complexity, the method's usage of euler angles in the orientation representation results in singularities.

The reconstruction of dense 3D indoor environments maps based on RGB-D SLAM was first proposed in [27]. Detection of features with the scale invariant feature transform [28] method extracts descriptors from two adjacent Red Green Blue (RGB) frames. Depth information is then added to generate 3D-3D feature point pairs information. RGB-D sensors, which provide both RGB images and depth information, have been integrated into SLAM systems to improve map reconstruction and pose estimation, as demonstrated in [29]. In this approach, the RGB image and depth data are utilized to generate a vertex map and a normal map, which are essential for creating accurate 3D representations of the environment. During the pose estimation step, the algorithm applies ICP alignment between the current surface and the predicted surface, which is generated based on data from the previous step. This step ensures the accurate alignment of successive frames in the system's reconstruction. Following this, the reconstruction update step integrates the newly acquired depth frame into the existing 3D model, improving the map's resolution

and accuracy. This method is particularly effective for mapping medium to large sized indoor environments, where dense and accurate 3D reconstructions are crucial for tasks such as navigation and object recognition.

2.1.3 Registration

Point cloud registration is essential in SLAM for aligning 3D scans and building coherent maps. It typically involves estimating rigid transformations by minimizing geometric projection error through alternating correspondence search and transformation estimation steps. With accurate correspondences, the transformation can be computed using a closed-form solution [30].

Point cloud registration methods, ranging from traditional geometric algorithms to recent deep learning-based approaches [30], vary in effectiveness depending on the application and data characteristics. Optimization-based methods, as described by Cheng et al. [31], typically follow an iterative process where initially inaccurate correspondences are refined over time, resulting in more accurate transformation estimates. These methods are mathematically grounded, offering convergence guarantees and requiring no training data, which allows them to generalize well to unseen environments. However, to address challenges such as noise, outliers, varying point cloud densities, and partial overlaps, they often depend on complex heuristics that can increase computational overhead and hinder real-time performance in large-scale or dynamic scenarios.

Common optimization-based techniques include ICP-based, graph-based, GMM-based, and SDP-based methods [30]. ICP variants iteratively minimize distances between closest points and are favored for their simplicity and speed, though they are sensitive to initialization and prone to local minima. Graph-based methods represent scans as nodes and optimize global consistency—useful in SLAM but computationally intensive. GMM-based approaches model point clouds as probabilistic distributions, enhancing robustness to noise and partial overlaps but requiring significant computation. SDP-based methods offer globally optimal solutions via convex formulations but struggle with scalability due to high computational costs.

In contrast, Yang et al. [32] introduced end-to-end learning-based registration, treating transformation estimation as a regression problem embedded within a neural network. These methods take two point clouds as input and output a transformation matrix, eliminating the need for explicit correspondence estimation. Some models focus on learning

global features, while others are designed to capture detailed correlations between source and target point sets. Huang [33] proposed a feature-metric registration approach that minimizes differences in learned features rather than geometric errors, combining deep learning with classical optimization for improved performance and robustness.

2.2 Background

In this subchapter some of the theoretical and mathematical backgrounds of VSLAM will be discussed.

2.2.1 Sensing Modalities in VSLAM

The selection of the sensor used by VSLAM systems is dependent on the particular application. Monocular, stereo, RGB-D cameras, and IMU are the main sensors used for VSLAM. A range of camera types and IMUs are primarily utilized to enable robots and devices to perceive and comprehend their environment through visual input and motion tracking.

For each camera in VSLAM setup we use camera calibration, which can be defined as intrinsic and extrinsic. The camera matrix [34] as expressed in equation 2.1 consists of the focal lengths and optical center, which are the intrinsic camera calibration parameters. The extrinsic camera parameters are the translation vector and rotation matrix of the camera with respect to the world frame or the reference frame, mostly the first camera pose. Projection matrix 2.2 combines both intrinsic and extrinsic parameters to map 3D world points to 2D image coordinates.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

$$P = K \begin{bmatrix} R & | & t \end{bmatrix} \quad (2.2)$$

In addition we need distortion of the pinhole camera can be described [34] as radial and tangential. The Radial distortion can be represented as follows:

$$(x_{\text{dis}}, y_{\text{dist}}) = (x(1 + k_1r^2 + k_2r^4 + k_3r^6), y(1 + k_1r^2 + k_2r^4 + k_3r^6)) \quad (2.3)$$

Tangential distortion occurs because the image-taking lens is not aligned perfectly parallel to the image plane. The amount of tangential distortion can be expressed as 2.4:

$$(x_{\text{dist}}, y_{\text{dist}}) = (x + (2p_1xy + p_2(r^2 + 2x^2)), y + (p_1(r^2 + 2y^2) + 2p_2xy)) \quad (2.4)$$

The distortion coefficients that we need for our camera setup can be written as 2.5:

$$\text{Dist\_coefficients} = (k_1, k_2, p_1, p_2, k_3) \quad (2.5)$$

Monocular Camera

Monocular camera with the pinhole model [35], consists lens and an image sensor. As shown in Figure 2.1, the monocular camera setup utilizes a single camera for capturing images. One fundamental limitation of monocular cameras is their inability to perceive the true scale of the world. Consequently, monocular SLAM can only estimate the map and camera trajectory up to an arbitrary scale. Additionally, scale drift can occur, causing different regions of the map to have inconsistent scales, especially in distant areas.



Figure 2.1: Monocular camera

Stereo Camera

A stereo camera system consists of two mounted cameras with a baseline distance. In stereo cameras the image capturing occurs simultaneously which is important for accurate depth estimation. Depth can be computed from a single stereo image pair by identifying corresponding points between the left and right images. By analyzing the disparity between these corresponding pixels, the system can estimate the 3D structure of the scene, enabling depth perception and improving spatial understanding in visual SLAM applications.



Figure 2.2: Stereo-ZED2 camera

RGBD

An RGB-D camera captures both RGB color and depth information using a single sensor system. Unlike stereo cameras that estimate depth through disparity, RGB-D cameras obtain depth data directly, typically using active sensing techniques such as structured light or time-of-flight. The RGB sensor records standard color images, while the depth sensor provides per-pixel distance measurements to objects in the scene. This direct acquisition of depth simplifies 3D reconstruction and object segmentation tasks, making RGB-D cameras particularly effective for visual SLAM, robotics, and augmented reality applications where accurate depth perception is essential.

IMU

In addition to the camera types mentioned above, an IMU is commonly used alongside cameras for visual-inertial navigation. As shown in Figure 2.3, the IMU measures an object's orientation, velocity, and gravitational forces. It typically comes as a 6-axis or 9-axis sensor, including a 3-axis accelerometer and a 3-axis gyroscope; 9-axis IMUs also include a 3-axis magnetometer. The accelerometer measures linear acceleration, the gyroscope measures angular velocity, and the magnetometer detects the Earth's magnetic field, which helps determine absolute orientation (i.e., heading relative to magnetic north)

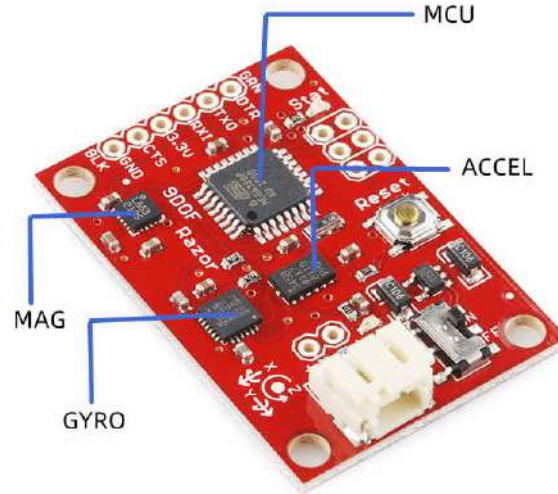


Figure 2.3: Inertial Measurement Unit

2.2.2 VSLAM and Visual-Inertial SLAM

Due to its lightweight nature and lower cost compared to other sensor modalities like LIDAR, VSLAM is widely adopted for estimating camera pose in real time and reconstructed environment. Using cameras, Visual SLAM systems track cameras to generate maps of the surrounding environment in an efficient manner. The affordability of cameras makes Visual SLAM particularly attractive for applications in robotics, drones, augmented reality (AR), and other autonomous systems, where cost-effective solutions are crucial without sacrificing performance. However, VSLAM, especially with monocular cameras, faces challenges in scale estimation and robustness. In environments with poor texture, motion blur, or dynamic elements that cause occlusions, these issues can lead to tracking errors and localization drift. The IMU provides high-frequency data on angular velocities, linear acceleration, and orientation, helping to improve pose estimation even under challenging conditions. By combining IMU and camera data, VSLAM resolves scale ambiguity in monocular systems, enabling more accurate localization and mapping in various environments. Visual-Inertial fusion SLAM is achieved by fusing data from an Inertial Measurement Unit and cameras. As shown in the simple diagram in Figure 2.4, the fusion of the IMU and camera is utilized for Visual SLAM.

Based on the camera input processing, computational methods, and the way the map is created VSLAM could be approached in a different way. The feature-based VSLAM approach significantly reduces computational complexity, making it widely adopted [37].

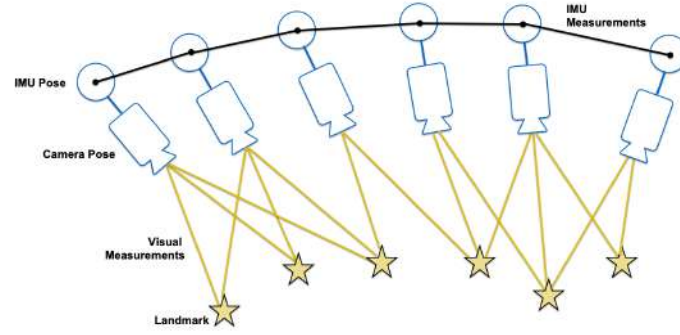


Figure 2.4: Visual-Inertial fusion [36]

This methods based on point features typically begin by applying feature detection algorithms such as SIFT [28], SURF [38], and ORB [39] to extract key points from images for matching. The camera pose is then estimated by minimizing the re-projection error. Feature points and their corresponding descriptors are utilized for data association, ensuring accurate tracking. Furthermore, during initialization, data association is achieved by matching feature descriptors [40]. Indirect VSLAM assumes brightness constancy, while direct VSLAM uses intensity gradients to minimize photometric error, unlike indirect VSLAM which minimizes re-projection error for accuracy. Indirect Visual SLAM splits the overall problem –recovering scene structure and camera poses from images - into two steps: first, a set of salient image features is extracted from the image and matched to other features in previous frames using invariant feature descriptors; second, based on this correspondence matching the camera motion and the scene geometry can be computed through triangulation, epipolar geometry, and the minimization of the re-projection error.

The re-projection error as shown in figure 2.5 measures how accurately a 3D point, when projected back into an image using a camera model, aligns with its observed 2D image point. It quantifies the difference between the actual pixel coordinates of a feature in the image and the pixel coordinates obtained by projecting the estimated 3D point using the camera parameters. Indirect method optimizations are based on minimizing the reprojection error which is a geometric error with good convergence properties.

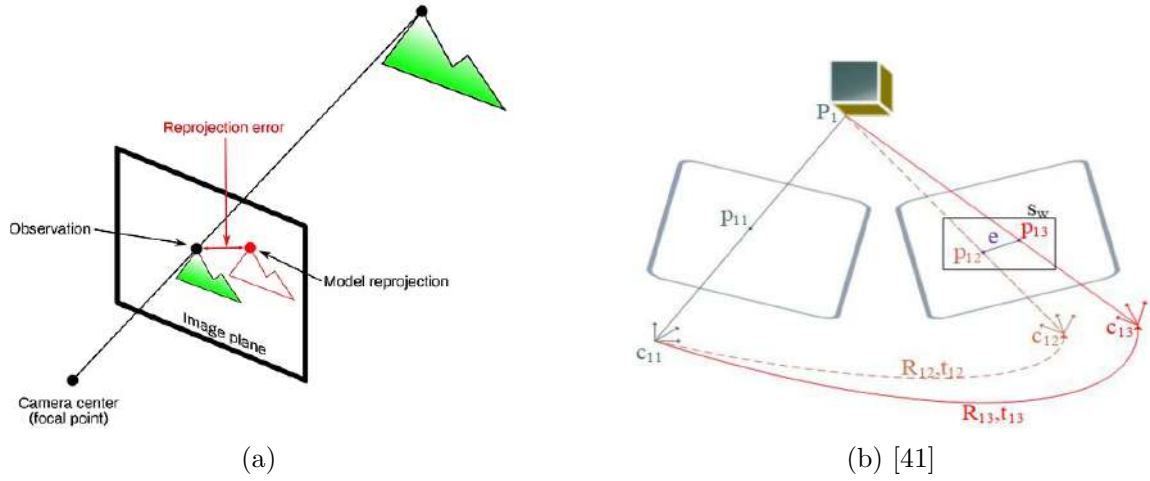


Figure 2.5: Visual comparison of re-projection error representations.

Mathematically the re-projection error is described as in eq 2.6

$$E = \sum_i \|p_i - \pi(K[R \mid t]P_i)\|^2 \quad (2.6)$$

where:

- E is the total re-projection error,
- p_i is the observed 2D image point,
- P_i is the corresponding 3D point in the world coordinate system,
- K is the camera intrinsic matrix,
- $[R \mid t]$ represents the camera extrinsic parameters (rotation R and translation t),
- $\pi(\cdot)$ is the projection function that maps a 3D point to 2D image coordinates.

Direct VSLAM uses the raw pixel intensity information from the camera without relying on explicit feature extraction [42]. The system estimates the camera's motion by directly comparing intensity values between consecutive frames without the feature extraction process. The direct method can maintain a good effect in the environment with out distinctive features [43]. The first direct method slam [44] enables tracking by matching input images with synthesized views from reconstructed maps. This approach allows for highly accurate and detailed environmental reconstruction. However, the high computational cost of data storage and processing limits its real-time execution to GPU-based systems. The basic principle of the direct method can be shown in figure 2.6. P1

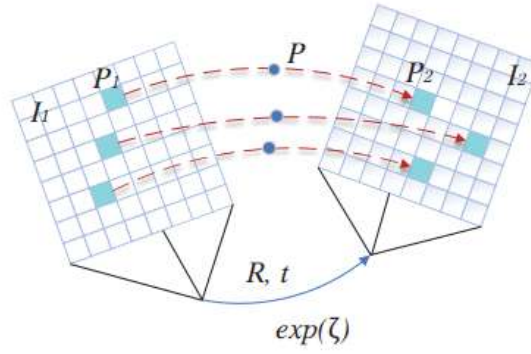


Figure 2.6: Direct visual SLAM principle [45]

and P_2 are the pixel coordinates of point P in the two images. Using the first camera as a reference frame, the rotation and translation of the second camera are R, t (the corresponding Lie group is T). The brightness error of two pixels of space point P is given by:

$$e = I_1(p_1) - I_2(p_2) \quad (2.7)$$

where e is a scalar. The optimization objective is to minimize the squared norm of the error:

$$\min_T J(T) = \|e\|^2 \quad (2.8)$$

If there are multiple points P_i in the space, the whole camera pose estimation problem becomes:

$$\min_T J(T) = \sum_{i=1}^N e_i^T e_i, \quad (2.9)$$

where

$$e_i = I_1(p_{1,i}) - I_2(p_{2,i}) \quad (2.10)$$

Chapter 3

Methodology

In this chapter an overview of both software and hardware components, along with the step-by-step procedures for VSLAM, are discussed. We detail the software and hardware specifications, the data collection approach, and the core algorithms pipelines implemented.

3.1 System Setup

3.1.1 Software Setup

For the software setup, we used ROS 2 [46] as the middleware on the Ubuntu operating system for our project. ROS 2 is a flexible framework that offers a collection of libraries and tools essential for the development of robotic applications. It provides an abstraction layer for hardware, allowing us to easily interface with various sensors and actuators. ROS 2 enables communication between different software components and provides an efficient mechanism for data exchange through topics, services, and actions.

One of the key advantages of ROS 2 is its real-time capabilities, enhanced security features, and improved scalability over its predecessor, ROS1. This made it a suitable choice for our VSLAM project, where timely data processing and communication between various modules are critical. In addition, ROS 2's extensive ecosystem of packages greatly simplified the implementation of algorithms like VSLAM and many others. We used ROS Visualization Tool 2 (RViz 2), which is a powerful visualization tool designed specifically for ROS 2. It is mainly used to display sensor data, robot models, and the robot's navigation state. RViz 2 provides a graphical interface where we can visualize the robot's trajectory, map, sensor readings, and other essential information in real-time.

3.1.2 Data Collection and Pre-processing

To collect real sensor data, we relied on ROS Data Recording and Playback Tool (ROSBag), a tool within the ROS ecosystem that allows recording and playing back ROS message data. ROSBag is a tool for collecting data streams from cameras and IMU. It captures all the messages published by the specified ROS 2 topics, storing them in a file format that can be later played back for our analysis or algorithm testing. This makes it particularly useful for simulating real-world environments and testing VSLAM algorithms without the need to collect live data repeatedly. In addition to using collected data in real time, we also used the well-known benchmark data sets TUM RGB-D to evaluate performance using the ground truth trajectory.

Table 3.1: Summary of datasets used for evaluation

Dataset	Number	Environment	Notes
Custom	3	2 Indoor, 1 Outdoor	Real sensor data recorded using ROSBag
TUM RGB-D Benchmark	7	Indoor	Public benchmark with ground truth trajectories

3.1.3 Hardware Setup

The HUNTER ground mobile wheeled robot, as shown in figure 3.1 with ZED camera, Ouster LiDAR, and IMU, was mainly used, and Unmanned Aerial Vehicle (UAV) with ZED cameras are used to collect the data and test our method.



(a) Mobile robot and drone platform used for data collection.



(b) Sensors used in the experimental setup.

Figure 3.1: Main Hardware Setup. Figure 3.1a shows the ground robot equipped with the drone used to collect ROS 2 bags. Figure 3.1b shows the main sensors integrated with the mobile ground robot.

The ZED camera, as shown in Figure 3.2, is developed by Stereolabs [47]. It is equipped with two high-definition RGB sensors that capture images with a field of view similar to human vision, allowing the ZED camera to create accurate stereo depth maps by analyzing the disparity between the two images. With stereo vision, the ZED camera captures images from each camera and compares corresponding points in these images to calculate the depth or distance of objects in the environment. The ZED camera can measure the depth of objects in its field of view with good precision. Additionally, the ZED camera is equipped with an IMU. We used mainly ZED2 for indoor cameras and ZEDX for outdoor cameras for data collection and testing. Although both are almost the same, there are some differences in their size, resolution, and maximum frame rate. The basic specifications for the ZED 2 camera are summarized in table 3.2.

Specification	Value
Sensor Type	Stereo camera
Resolution (Max)	2208×1242 @ 15 fps
Pixel Size	$2 \mu\text{m} \times 2 \mu\text{m}$
Field of View	$110^\circ(\text{H}) \times 70^\circ(\text{V}) \times 120^\circ(\text{D})$
Baseline	0.12 m
IMU	3-axis accelerometer, gyroscope and magnetometer

Table 3.2: Basic Specifications of the ZED 2 Camera [47]



Figure 3.2: ZED Camera[47], sensor used for data collection.

3.2 Main Pipeline Features Based (Indirect) VSLAM

The process of solving VSLAM systems can be divided into two key components: *Front-End* and *Back-End*, as shown in Figure 3.3. In addition, depending on how image data is processed for pose estimation, VSLAM can be classified into *direct* and *indirect* methods. *Indirect VSLAM* relies on data from a camera—such as monocular, stereo, or RGB-D—to detect and track distinct features in the scene. These features are then used to estimate the sensor’s position and construct a map of the environment. On the other hand, *direct VSLAM* does not require the extraction of distinct features. Instead, it operates directly on the image intensities, leveraging photometric consistency between frames to estimate motion and structure.

Visual-inertial SLAM enhances VSLAM by integrating data from an inertial measurement unit alongside visual input. The IMU provides acceleration and angular velocity measurements, improving motion estimation, particularly in challenging scenarios such as blurred motion, featureless environments, or rapid movements, which enhances robustness and accuracy in state estimation.

- **Front-End (Perception and Data Association):** This stage involves processing sensor data, extracting key features, and estimating motion. It includes tasks such as feature detection, feature tracking, or feature matching, visual odometry, and loop closure detection to track the robot’s movement and recognize previously visited locations.
- **Back-End (Optimization and Map Refinement):** This stage focuses on refining the estimated trajectory and map by minimizing errors. It employs techniques including BA and PGO to improve consistency, reduce drift, and enhance overall VSLAM accuracy.

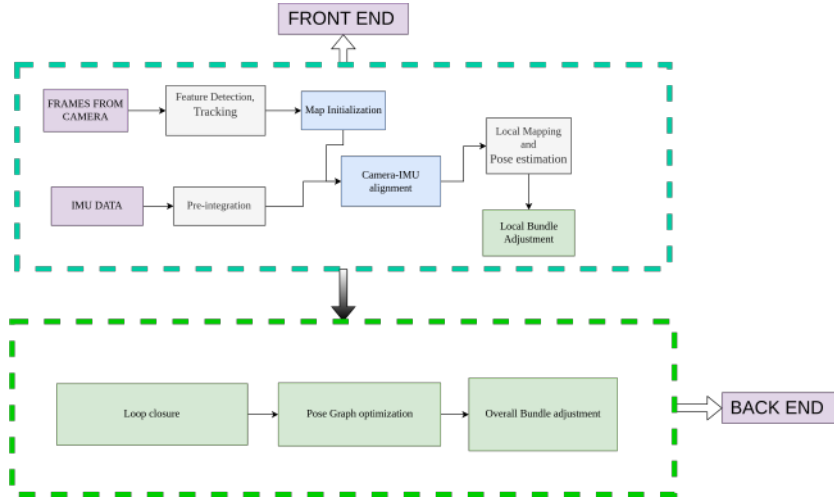


Figure 3.3: General VSLAM diagram for both the Front-End and Back-End components

As illustrated in Figure 3.4, the proposed system is based on [18]. The pipeline has been implemented within the ROS2 framework to enable modular, real-time operation, and integration with other robotic components.

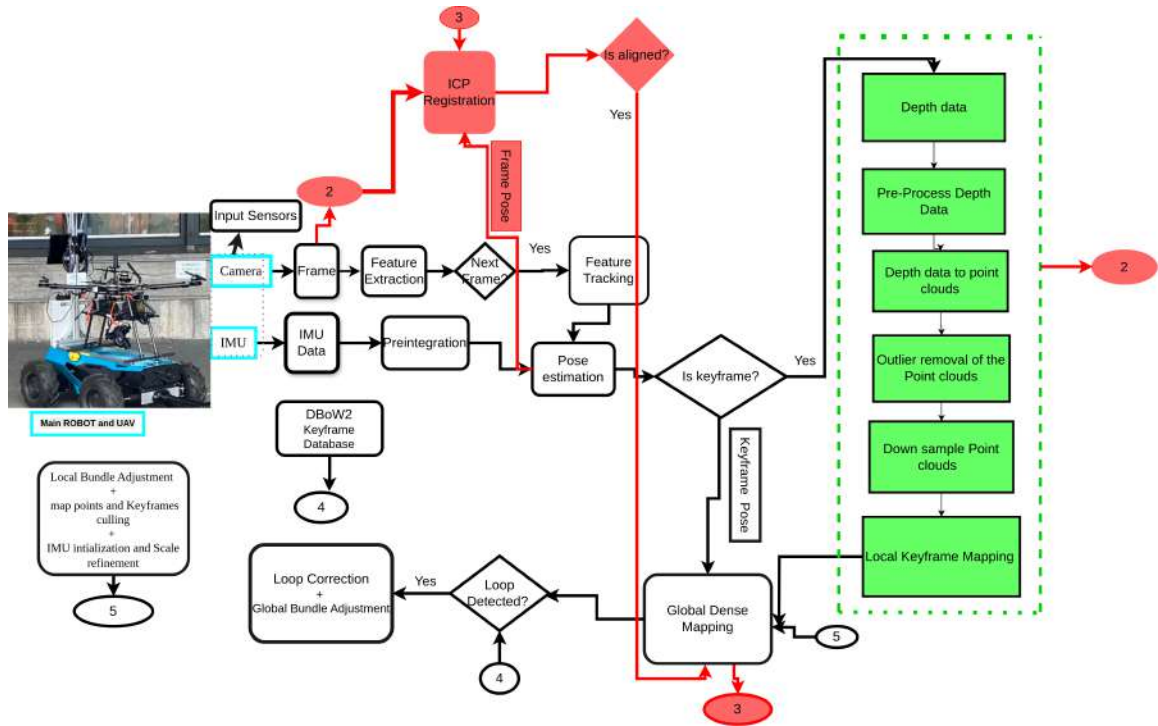


Figure 3.4: Proposed method Pipeline: (2) $\rightarrow$ represents when there is a new frame from the input camera sensor, the frame is processed to give the dense local map which uses source to ICP, where as (3) $\rightarrow$ uses as the previous frames aligned dense map are used as a target to the ICP registration. The (4) and (5) $\rightarrow$ are a collection of DBoW vectors for each keyframes which are used to compare with the current keyframe for loop closure detection, and for optimization until the current keyframe respectively.

Most methods (white boxes) are adapted from [18].

As shown in Figure 3.4, the main proposed method is illustrated using numerical representations to keep the diagram compact and informative. The diagram distinguishes between two approaches using color coding: red indicates the method mainly for indoor environments, while green represents the approach that is robust for both indoor and outdoor scenarios.

3.2.1 Feature Detection

Feature-based VSLAM identifies distinctive image elements like corners or patterns to different frames. Images are first converted to grayscale for efficiency, then features are extracted using algorithms such as Harris, Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), or ORB. ORB is favored for real-time use due to its efficiency, robustness, and rotation invariance. It detects keypoints with FAST, refines them using the Harris measure, and employs an image pyramid for scale robustness, making it effective in varying conditions and low-power devices. Feature detection examples are shown in Figures 3.5a and 3.5b.

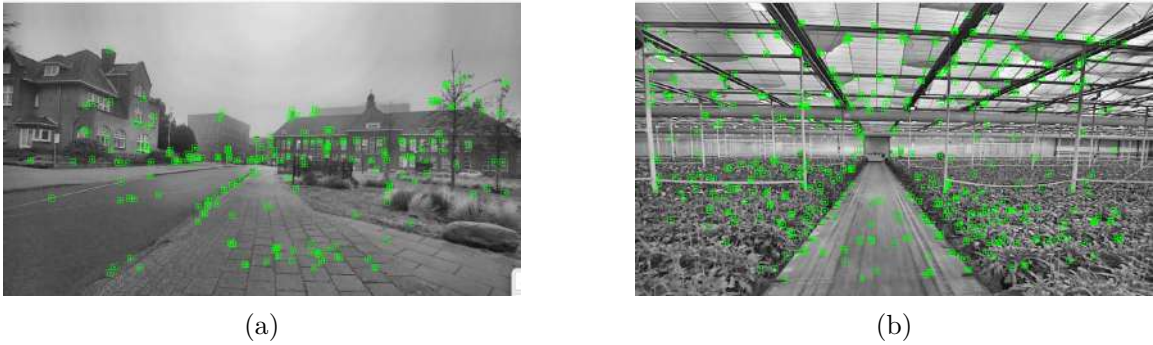


Figure 3.5: ORB feature detection in different environments: outdoor (3.5a) and indoor (3.5b) scenes.

3.2.2 Feature Matching and Tracking

After features are detected, there are two possible ways to obtain the relative poses of the camera across a sequence of frames. Similar to the first frame, keypoints in the next frame are detected using the FAST corner detector, and BRIEF descriptors are computed. These descriptors are binary strings that represent the local image patch around each keypoint. ORB features are then matched between consecutive frames to estimate the motion of the camera. This is achieved by comparing the BRIEF descriptors and finding

correspondences between keypoints in the current frame and those in the map or the previous frame.

Feature tracking involves detecting features in the first frame and then following their movement in subsequent frames using local search techniques. The most common approach for this is the Kanade-Lucas-Tomasi Tracker (KLT), which minimizes the photometric error between frames. KLT considers that the appearance of a small patch around a feature remains constant as it moves from one frame to another. KLT works based on the following assumptions; first, it assumes that optical flow is locally smooth, meaning that the motion of objects or points within a small, localized region of an image remains relatively consistent without abrupt changes. Second, it assumes that neighboring pixels exhibit similar displacement, allowing the algorithm to approximate the overall motion of the region using a single motion vector. The movement of a feature is modeled as a two-dimensional translation, making the optical flow estimation computationally efficient. It takes a 3×3 patch around a point, assuming that all 9 points exhibit the same motion. The image gradients can be computed (f_x, f_y, f_t) for these 9 points [48]. This results in a system of 9 equations with two unknown variables, making it an overdetermined problem. A better solution is obtained using the least squares method. The final solution, which involves solving a two-equation, two-unknown system, is given as in equation 3.1.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum f_x^2 & \sum f_x f_y \\ \sum f_x f_y & \sum f_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum f_x f_t \\ -\sum f_y f_t \end{bmatrix} \quad (3.1)$$

The algorithm tracks a given set of points by computing optical flow vectors. However, it struggles with large motions, as it is designed for small displacements. To overcome this, image pyramids are used to break down large motions into smaller ones, enabling robust optical flow estimation at multiple scales. However, feature matching as shown in figure 3.7 involves independently detecting features in each frame and finding correspondences based on their descriptors.

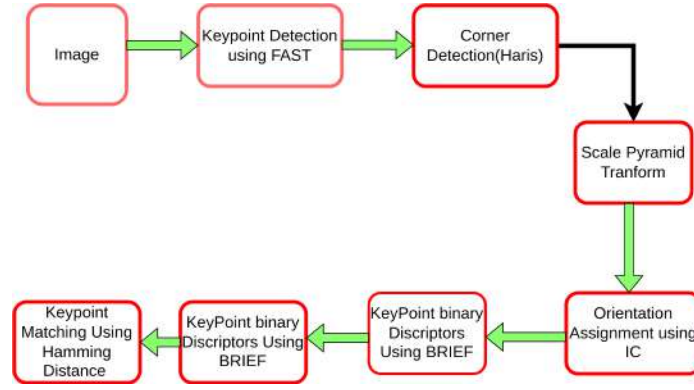


Figure 3.6: ORB feature detection and matching pipeline.

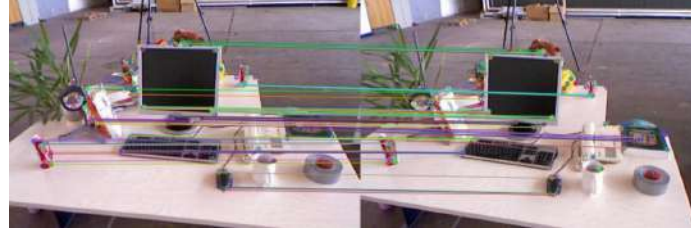


Figure 3.7: Feature Matching [49].

3.2.3 RANSAC

After feature matching and tracking, outliers can affect motion estimation and 3D reconstruction. To eliminate the outliers, the RANSAC algorithm is used. It repeatedly selects random subsets of data to fit a model and evaluates how well the full dataset fits that model. This process continues until the best-fitting model with minimal outliers is found, as outlined in Algorithm 1.

Algorithm 1 General RANSAC Algorithm

- 1: Randomly choose n samples. Where, n is the minimum number of samples required to fit a model.
 - 2: Fit the model to the randomly chosen samples.
 - 3: Count the number M of data points (inliers) that fit the model within a measure of error ϵ .
 - 4: Repeat Steps 1–3 N times.
 - 5: Choose the model that has the largest number M of inliers.
-

3.2.4 Pose Estimation

After the features are matched and tracked, and outliers are removed, pose estimation is performed using the Fundamental Matrix (FM) or Essential Matrix (EM), depending

on whether the camera's intrinsic parameters are known.

The FM encapsulates the epipolar geometry between two views and is computed from corresponding feature points. It satisfies the epipolar constraint as in equation 3.2.

$$x_2^T F x_1 = 0 \quad (3.2)$$

where x_1 and x_2 are corresponding points in two images. Once F is estimated, it can be decomposed to obtain EM if intrinsic camera parameters are available.

The EM is related to the intrinsic matrix of the camera K and the fundamental matrix as in equation 3.3, assuming the same camera is used.

$$E = K^T F K \quad (3.3)$$

It contains information about the **relative rotation** R and **translation** t between two views, extracted using singular value decomposition.

3.2.5 IMU Pre-integration

IMU is integrated into VSLAM to enhance motion estimation, especially in low-texture or low light environments. It provides high-frequency acceleration and angular velocity data, which help predict motion between frames. Using *preintegration*, IMU data are accumulated over short intervals between frames to estimate relative motion efficiently, reducing computational cost. However, IMU measurements are affected by biases, mainly from the accelerometer and gyroscope, which must be estimated and corrected. Biases can be static, temperature-dependent, or unstable over time. Static bias can be averaged out when the device is stationary. Temperature-related bias can be modeled using linear or quadratic functions of the IMU's temperature and subtracted accordingly for more accurate motion tracking.

Bias instability refers to the slowly changing bias of the sensor over time due to internal noise and aging effects. This causes random walk and drift, leading to accumulating errors in motion estimation. Without compensation, this drift can significantly degrade the accuracy of VSLAM systems over time.

Gyroscope bias instability causes orientation drift because even a small bias in angular

velocity accumulates over time when integrated as in equation 3.4.

$$\theta(t) = \int_0^t (\hat{\omega} - b_g) dt \quad (3.4)$$

If the bias b_g drifts over time, the error in orientation grows quadratically and can be estimated as in equation 3.5.

$$\theta_{\text{error}}(t) \approx \frac{1}{2} b_g t^2 \quad (3.5)$$

This means that even a small bias instability in the gyroscope can result in large angular drift over long periods.

For an accelerometer, bias instability leads to errors in velocity and position when it is integrated with time, which can be modeled as equations 3.6 and 3.7, respectively.

$$v(t) = \int_0^t (\hat{a} - b_a) dt \quad (3.6)$$

$$p(t) = \int_0^t v(t) dt \quad (3.7)$$

If the bias b_a drifts over time, the position error grows cubically in equation 3.8.

$$p_{\text{error}}(t) \approx \frac{1}{6} b_a t^3 \quad (3.8)$$

The relative motion resulted from the pre-integration of the IMU measurements between two frames is combined with feature-based visual odometry using an optimization framework, which improves robustness, particularly in rapid motion or low-texture scenes. In addition, IMU data help to estimate the gravity vector, which helps to correct for drift and maintain orientation stability over time [18].

3.2.6 Visual Inertial Fusion

In visual-inertial SLAM, accurate state estimation requires precise alignment between the camera and the IMU. Due to differences in their timing and spatial placement, both temporal and spatial calibration are essential to ensure effective data fusion between the two sensors.

In ORB-SLAM3 [18] addresses this by estimating the spatio-temporal calibration between the sensors. The process includes determining the relative pose (translation and rotation) between the camera and IMU, as well as the time offset between their mea-

surements. Accurate alignment ensures that the visual and inertial data are consistent, enabling the system to leverage the complementary strengths of both sensors. As the IMU provides high-frequency motion estimates that can compensate for the lower frame rate of the camera, while the camera provides precise positional information that corrects for the drift inherent in IMU measurements. This alignment process is typically performed during system initialization and refined during operation to maintain accuracy using optimization.

Extrinsically, the IMU and camera are in different positions and can be calibrated using a tool such as Kalib [50], [51], [52], [53], and [54], which estimates the spatial and temporal parameters of the camera system with respect to an intrinsically calibrated IMU. The fusion of IMU and camera data in visual-inertial SLAM is achieved through a tightly coupled pose graph optimization. Each node represents a keyframe pose, while edges encode constraints from visual reprojection and preintegrated IMU measurements. Visual constraints minimize the reprojection error of 3D points, and inertial constraints capture motion between keyframes. Joint optimization refines poses, velocities, and IMU biases by minimizing both error types. This fusion leverages the strengths of both sensors: the camera corrects IMU drift, and the IMU ensures robust tracking during rapid motion or poor visual conditions, enabling accurate and consistent localization [18].

3.2.7 Mapping

Mapping is a fundamental component of the VSLAM problem for constructing and refining a representation of the environment as the system moves through it.

In VSLAM, the map initialization depends on the camera type that is used; with a single camera, depth information is not directly available from a single frame. Therefore, monocular SLAM requires motion between at least two frames to perform triangulation and estimate 3D positions of features. This makes the initialization process more complex and sensitive, often involving structure-from-motion techniques to build an initial sparse map. In stereo systems, use a pair of cameras with a known baseline to estimate depth via stereo matching. The depth information can be obtained directly from the disparity between the two images, which greatly simplifies the initialization process.

In [18], this is achieved using a combination of visual and inertial data. The system first detects and matches ORB features between two consecutive frames to estimate the relative pose of the camera. If sufficient parallax (displacement of features between frames)

is observed, the system triangulates the 3D positions of these features to construct an initial sparse map. The IMU data is then used to refine this initial estimate by providing additional constraints on the camera's motion. This hybrid approach ensures that the map is accurately initialized even in challenging conditions, such as when the camera undergoes rapid motion or when the environment lacks distinctive visual features. Once the initial map is created, tracking the camera's pose relative to this map continues to expand it as the sensor moves through the environment. However, for RGB-D cameras, a direct 3D map can be created from a single frame, as depth information is available along with the RGB image. This enables immediate and highly accurate initialization, even without any camera movement. Generally, mapping in visual SLAM has two possible stages to get local and global mapping, as described below.

3.2.8 Local and Global Mapping

Local mapping refers to the process of constructing a map of the environment in the local coordinate system based on the motion of the camera, allowing the system to continuously update and refine the map while tracking the pose of the camera. It is based on the frame of reference of the camera, and camera poses are used to transfer the local map to the global map. Local mapping is essential to maintain the consistency and accuracy of the map, especially as the camera moves through dynamic environments or encounters features with limited visibility. Local mapping is simplified by triangulating the features tracked in the images, as shown in Figure 3.8. A local map built using dual-camera visual odometry during exploration is shown, with map points of the forward camera marked in red and those of the downward camera marked in blue [55].

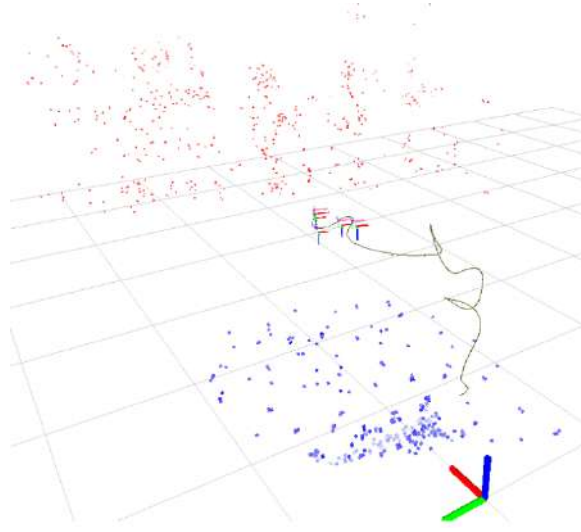


Figure 3.8: Local mapping [55]

To create a dense map of the environment, an RGB-D camera within a ROS2-based ecosystem is used. The RGB-D camera captures both color (RGB) and depth information, which is essential to generate 3D representations. This pipeline, as shown in Figure 3.9, ensures that the mapping process is both efficient and reliable, making it suitable for creating a global map, which is the accumulation of local maps.

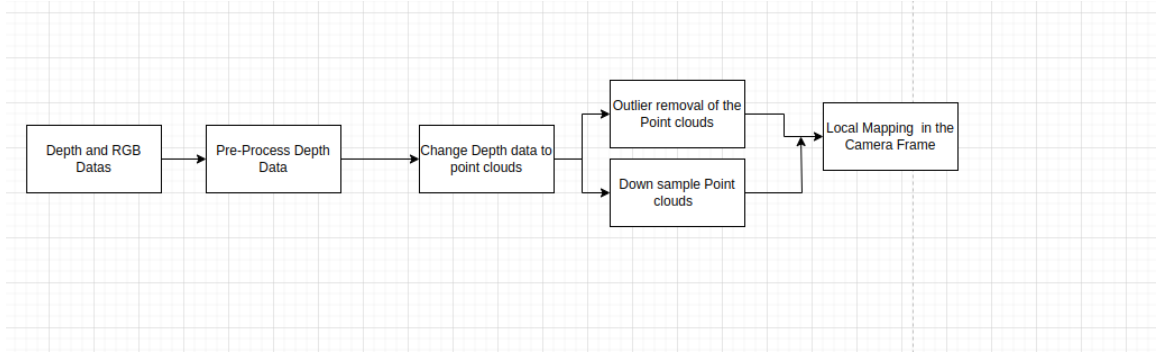


Figure 3.9: RGB-D camera based local mapping pipeline.

The process consists of the following steps:

- **Getting Depth Data:** The depth data is extracted from the RGB-D camera's sensor. This data provides information about the distance of objects from the camera.
- **Pre-processing Depth Data:** Before using the depth information, we apply filtering techniques such as noise reduction, depth smoothing, and outlier removal. These preprocessing steps help enhance accuracy and remove inconsistencies caused by sensor noise or environmental factors.

- **Conversion to Point Cloud:** The refined depth data is converted into a 3D point cloud using the Point Cloud Library (PCL). This transformation allows for a structured representation of the environment in three dimensions.
- **Statistical Outlier Removal:** This is a filtering technique used to clean point cloud data by removing noisy or spurious measurements. It operates by analyzing the distribution of distances between each point and its neighbors. Points whose average distance to their neighbors significantly deviates from the mean are considered outliers and are removed.
- **Downsampling:** It is a method used to reduce the density of a point cloud while preserving its overall structure using the voxel grid. The space is divided into a 3D grid of equally sized cubes, known as voxels. For each voxel, the points within it are approximated by a single representative point—typically the centroid.
- **Local Mapping:** The generated point cloud is used to build a local map. Each keyframe in the mapping process corresponds to a specific camera pose, and the accumulated keyframes help form a dense and accurate representation of the surrounding space.

Point clouds have become a key format for 3D scene representation. Due to their limited field of view, sensors like camera and LiDAR capture only a partial view of the environment, making point cloud registration essential. Point cloud registration is the process of estimating the transformation matrix, as in figure 3.10, that aligns two or more point cloud scans [56]. By applying these transformation matrices, we can seamlessly merge multiple partial scans of the same object or scene into a unified and complete 3D model.

The formulation of the point cloud registration problem can be expressed as follows:

Let $\mathbf{A} = [\mathbf{A}_1^T, \mathbf{A}_2^T, \dots, \mathbf{A}_M^T]^T \in \mathbb{R}^{M \times 3}$ and $\mathbf{B} = [\mathbf{B}_1^T, \mathbf{B}_2^T, \dots, \mathbf{B}_N^T]^T \in \mathbb{R}^{N \times 3}$ be two point clouds, where $\mathbf{A}_i, \mathbf{B}_j \in \mathbb{R}^3$ represent the coordinates of the i -th and j -th points in the respective point clouds

Suppose $\mathbf{A}$ and $\mathbf{B}$ have K pairs of correspondences. The goal of registration is to find the rigid transformation parameters f (rotation matrix $R \in SO(3)$ and translation vector $t \in \mathbb{R}^3$) which best aligns the point cloud $\mathbf{A}$ to $\mathbf{B}$ as shown below:

$$\arg \min_{R \in SO(3), t \in \mathbb{R}^3} \|d(\mathbf{A}, f(\mathbf{B}))\|_2^2 \quad (3.9)$$

where

$$d(\mathbf{A}, f(\mathbf{B})) = d(\mathbf{A}, R\mathbf{B} + t) = \sum_{k=1}^K \|\mathbf{A}_k - (R\mathbf{B}_k + t)\|_2^2 \quad (3.10)$$

is the projection error between $\mathbf{A}_k$ and transformed $\mathbf{B}_k$ ($k \in [1, K]$). The Equation 3.9 the optimal transformation matrix can be calculated if the true correspondences are known [57] [58]; in addition, correspondences can also be found if the optimal transformation matrix is given.

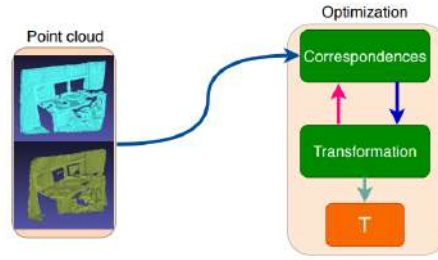


Figure 3.10: ICP Based Point cloud registration [56]

3.2.9 Bag of Words

DBoW is used for indexing and converting images into a bag-of-words representation[14]. Implements a hierarchical tree to approximate nearest neighbors in the feature space of the image and create a visual vocabulary. It has an image database using the inverted file structure to index images and to enhance quick queries. In the domain of 2D image processing, the bag-of-words Bag of Words (BOW) model [14] is commonly used for place recognition by extracting and retrieving 2D features such as SIFT [28] and ORB [59]. Specifically, BoW-based indexing for binary ORB features has proven effective in real-time place recognition. The approach constructs a visual vocabulary as a tree structure in an offline phase, using a set of descriptors extracted from a training image dataset. When processing a new image, the vocabulary transforms its extracted features into a low-dimensional representation based on the term frequency-inverse document frequency (tf-idf) score [60]. The tf-idf helps weigh visual words (ORB descriptors) in a way that common features are downweighted (less informative), and rare but repeatable features are upweighted (more informative) and score is computed as in the equation of 3.11.

$$tf-idf = \frac{n_{wi}}{n_i} \log \frac{N}{n_w}, \quad (3.11)$$

where n_{wi} represents the occurrence of word w in image l_i , and n_i denotes the total number of words in l_i . The parameter N is the total number of images processed, while n_w represents the number of images containing word w . The inverse document frequency (idf) component of tf-idf ensures that words appearing in fewer images receive higher scores, thereby increasing their discriminative power.

A higher tf-idf score indicates that a word is more frequent in the image but less common throughout the data set, making it more distinctive. If a word's score surpasses a certain threshold, the system calculates its similarity with words stored in the database. The inverted index is then used to efficiently retrieve matching images based on similar words.

3.2.10 Pose Graph Optimization and Bundle Adjustment

PGO is used to refine camera poses by modeling them as nodes in a graph, with edges representing spatial constraints between these camera poses which could be visual odometry or loop closures. It minimizes a global error function to adjust these camera poses, improving overall accuracy and consistency in the estimated trajectory.

As an example pose graph can be illustrated with 5 nodes as shown in figure 3.11, where $\mathbf{x} = (x_1, \dots, x_N)^T$ be a vector consisting of nodes $x_1, \dots, x_N$, where x_i for $i = 1, \dots, N$ is the camera pose at i . Let z_{ij} and E_{ij} be an edge that connects between the i and j nodes.

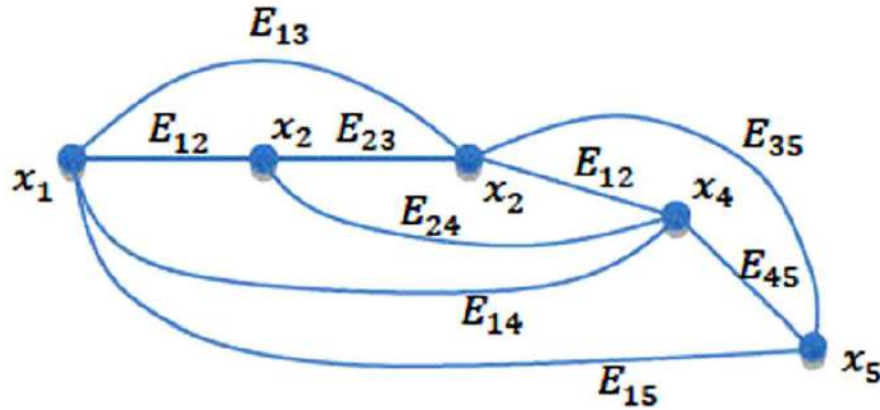


Figure 3.11: Pose graph [61]

BA refines both map points and camera poses by minimizing reprojection error through non-linear least-squares optimization. Given a set of 3D points and camera poses (each defined by rotation and translation), BA helps to align predicted and observed feature

points, as formulated in Equation 3.12.

$$\min_{\mathbf{P}_j, \mathbf{X}_i} \sum_{i=1}^N \sum_{j \in \mathcal{V}(i)} \|\mathbf{p}_{ij} - \pi(\mathbf{P}_j, \mathbf{X}_i)\|^2, \quad (3.12)$$

where:

- $\mathbf{p}_{ij} \in \mathbb{R}^2$ is the observed image projection of 3D point $\mathbf{X}_i$ in camera j ,
- $\pi(\mathbf{P}_j, \mathbf{X}_i)$ represents the projection function that transforms the 3D point into the 2D image plane using the camera intrinsic parameters,
- $\mathcal{V}(i)$ denotes the set of cameras observing the 3D point $\mathbf{X}_i$.

The function $\pi(\mathbf{P}_j, \mathbf{X}_i)$ follows the standard pinhole camera model:

$$\pi(\mathbf{P}_j, \mathbf{X}_i) = \mathbf{K} (\mathbf{R}_j \mathbf{X}_i + \mathbf{t}_j), \quad (3.13)$$

where $\mathbf{K}$ is the camera intrinsic matrix. It can also integrate bundle adjustment to refine keyframe-based maps like [18]. In multi-sensor fusion setups, bundle adjustment can also be extended to optimize additional parameters such as IMU biases and extrinsic calibration.

In summary, both BA and PGO are used to refine camera poses and, in the case of BA, also 3D map points. Local BA is used to optimize the poses of recent keyframes along with their associated map points within a limited time or spatial window, ensuring accurate and consistent tracking during mapping. Globally, it typically happens when there is loop closure. Global PGO is used to correct drift by adjusting all keyframe poses in the graph using the loop closure constraints, which is efficient and robust for large-scale corrections. Global BA, on the other hand, jointly optimizes all camera poses and 3D map points across the entire map for the highest level of global consistency, although it is computationally more expensive.

Chapter 4

Results and Discussions

In this chapter, we presented the results obtained from various experimental scenarios. The proposed algorithm was tested using data collected by the robots (figure 3.1a in both indoor and outdoor environments). Additionally, the algorithm was tested on publicly available benchmark datasets and evaluated on the TUM RGB-D benchmark dataset, which provides ground truth trajectory to measure accuracy. Although there is no ground truth for dense mapping, qualitative evaluations were conducted to assess the accuracy of the reconstructed maps. For implementation, a C++-based ROS 2 framework is used, and the resulting trajectories and their associated errors are visualized using the tool presented in [62].

4.1 Mapping and Localization with Custom Data Collected

In this section, we tested mapping and localization experiments conducted using ROS 2 bag files collected with the hardware setup described in Chapter 3. The data was gathered in various custom-designed environments around the lab, including both indoor and outdoor settings. These experiments were carried out to evaluate the system's performance in diverse and realistic scenarios.

4.1.1 Indoor Experiment

To evaluate the proposed method, the first experiment was carried out in an indoor environment. We used PCL to convert the raw depth data into point clouds, and, once

converted, local maps were created with respect to the camera frame, as shown in Figure 4.1b. In this figure, the depth data are converted to an RGB8-format point cloud and compared with the grayscale 4.1a).

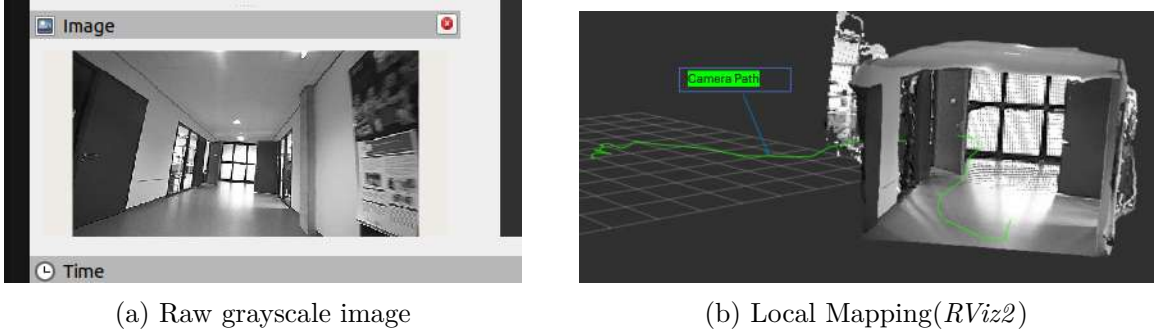


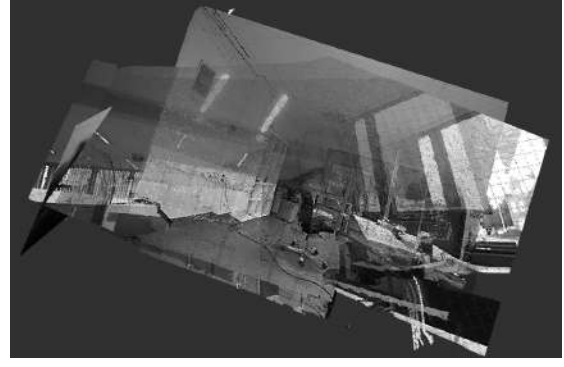
Figure 4.1: Comparison between the frame's grayscale image and the reconstructed local mapping: the green line indicates the camera path, and the resulting map is the frame's local mapping.

To align point clouds across frame sequences, we used ICP for registration, initializing with estimated transformation poses. Camera poses then transform the point clouds into the world frame to build a global map. We tested scenarios using the full camera view and cropped views, retaining key environmental features. To manage the dense point clouds, we applied voxel grid downsampling and removed noise using statistical outlier filtering from the PCL library.

The results, where point clouds were generated using the full field of view of the camera, are illustrated in Figures 4.2a and 4.2b. In Figure 4.2a, the camera was slowly moved to evaluate the effectiveness of ICP, resulting in well-aligned point clouds for two consecutive frames after the depth data of each frame was converted into point clouds. In contrast, in Figure 4.2b, the camera experienced fast and curved movements, and due to inconsistencies in the point clouds, the quality of alignment is significantly degraded in this case.



(a) ICP alignment of frame scans.



(b) Fast and curved camera movements.

Figure 4.2: ICP alignment using consecutive frames captured with an RGB-D camera over the full field of view.

To enhance the alignment and visualization of point clouds, we evaluated our method by limiting the field of view and applying ICP registration. We used point-to-point ICP, but it did not yield satisfactory results. After conducting several tests with point-to-point registration, we switched to point-to-plane ICP, which is generally more suitable for RGB-D-based point clouds because it considers local surface normals, leading to better convergence and alignment. This approach provided improved results, but its effectiveness was still limited to short trajectories (approximately 15 meters to indoor environments), as shown in Figures 4.3, 4.4, and 4.5. Since RGB-D cameras produce very dense point clouds for each frame, ICP registration becomes time-consuming, and the poses of the camera arrive with a delay. As a result, consecutive scans overlap less, leading to a lower ICP fitness score and reduced overall alignment accuracy.

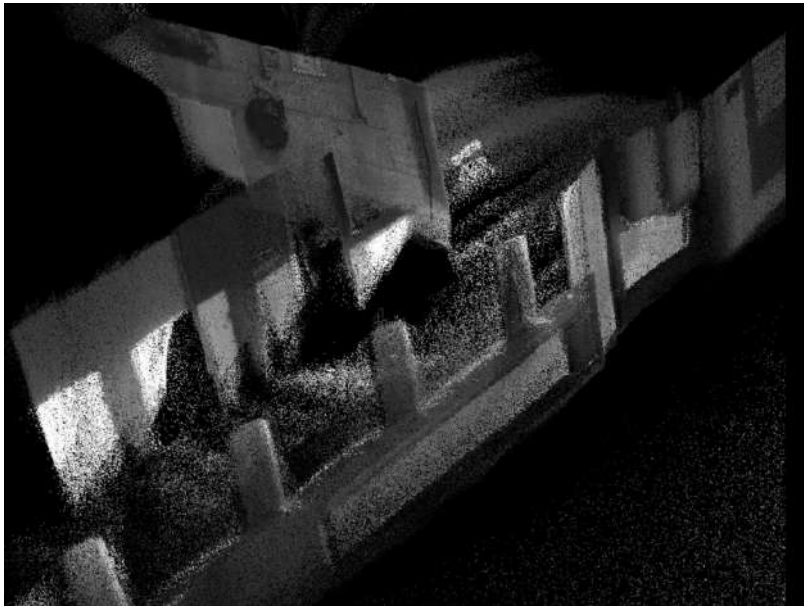


Figure 4.3: Dense map of small trajectories in an indoor environment using ICP, reconstructed with a limited camera field of view.



Figure 4.4: Reconstructed map showing increased trajectory length and extended vertical field of view, as a continuation of Figure 4.3.

The reconstructed 3D maps corresponding to Figures 4.3 and 4.4 are based on the environment shown in the raw image 4.5 in the right.

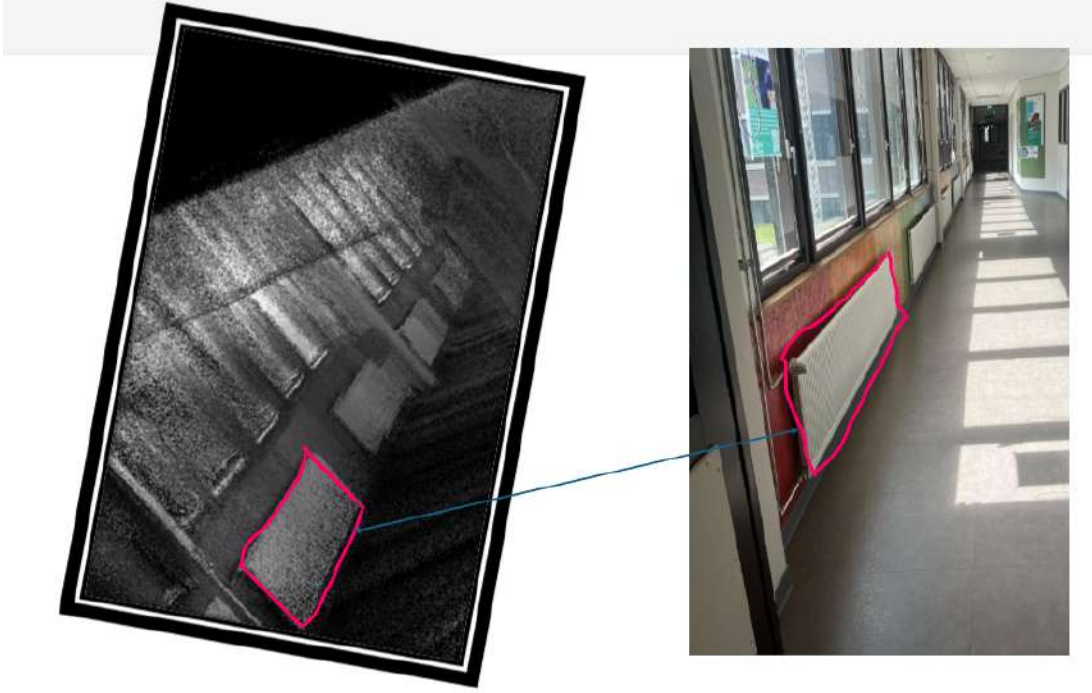


Figure 4.5: Visualization for qualitative comparison between the reconstructed dense map and the corresponding raw image. The view is zoomed in to highlight details: *left* — *accumulated dense map*; *right* — *raw image of the environment*.

We evaluated an additional indoor scenario using data from a handheld camera recorded in ROS 2 bag files. To overcome limitations of frame-by-frame depth acquisition and ICP-based alignment, we adopted a modified approach: dense mapping is performed only at selected keyframes (see Figure 3.4, green), skipping less informative frames. Most processing steps remain unchanged, except point cloud registration is not applied, and redundant dense points at keyframes are removed using the method from [18]. The resulting dense map is shown from two perspectives in Figures 4.6 and 4.7, with a zoomed view provided in Figure 4.7 for clarity.



Figure 4.6: Reconstructed dense map of the indoor scene from a side view for an indoor experiment, generated using the method proposed in Diagram 3.4(green).



Figure 4.7: Zoomed-in side view of the indoor scene from Figure 4.6, highlighting structural details for improved visualization.

In summary, the reconstructed map performed well in the indoor environment, as it lacked common challenges such as large featureless areas, repetitive textures, or dynamic

elements, factors that typically hinder reliable tracking and accurate mapping in VSLAM systems. Furthermore, the camera depth estimation is limited to a maximum range of 20 meters, making it particularly well suited and robust for indoor applications.

4.1.2 Outdoor Experiment

In addition to the indoor environment, an outdoor experiment was conducted to evaluate the robustness of the system under more challenging conditions. In this experiment, the robot followed a predefined path around the laboratory, as shown in Figure 4.8a (Google Maps). The corresponding dense map and trajectory results are shown in Figure 4.8b. For better visualization, the dense map is also displayed from multiple zoomed-in perspectives, as illustrated in Figure 4.9. Due to limitations of the field of view of the camera and the laptop's computational resources and the effective range of depth estimation, the resulting dense map does not cover the full field of view of the camera.



(a) Predefined path followed by the robot on Google Maps.



(b) Reconstructed dense map and estimated trajectory.

Figure 4.8: Comparison of the predefined GPS path and the reconstructed dense map with estimated trajectory.



Figure 4.9: Zoomed-in view of the reconstructed dense map for improved visualization

The estimated trajectory is analyzed and visualized, showing minimal variation along the Y-axis, as shown in Figures 4.10 (3D trajectory) and 4.11 (axis-wise position over time). This limited vertical movement is due to the camera being rigidly mounted on the robot. However, slight road irregularities introduce minor Y-axis deviations relative to the robot's starting position. In contrast, the X and Z axes exhibit more significant positional changes.

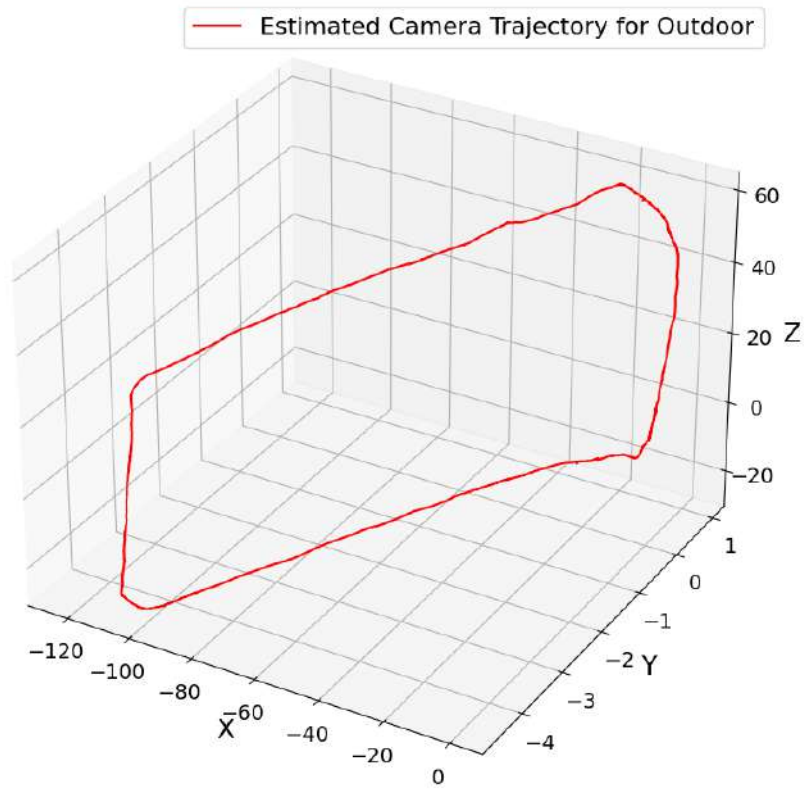


Figure 4.10: Estimated 3D trajectory of the robot in the outdoor environment.

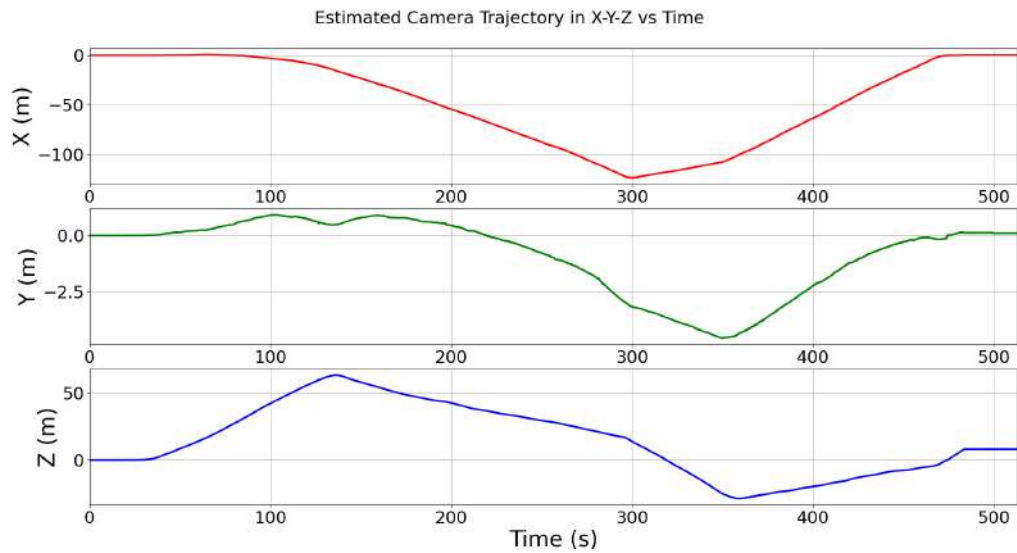


Figure 4.11: Estimated trajectories along the XYZ, showing pose evolution with respect to each reference axis.

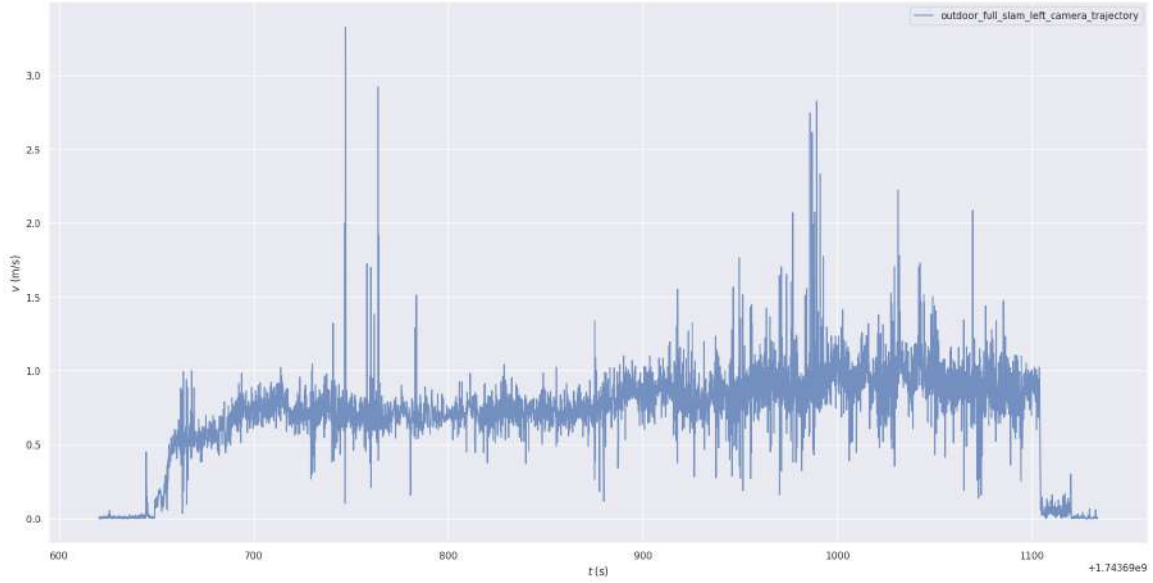


Figure 4.12: Robot speed over time for the entire trajectory, illustrating velocity variations at each time step throughout the outdoor movement experiment

The main challenges in the outdoor environment experiment are lighting variations, long-range depth inaccuracies, and reduced feature density. In addition, many objects are movable, making feature tracking less accurate due to significant differences between consecutive frames. Additionally, the lack of distinct features in road structures limits reliable feature detection, further complicating outdoor scene analysis.

4.2 Benchmark Dataset-Based Evaluation

We evaluated the method using sequences from publicly available benchmark datasets, mainly the TUM RGB-D dataset [63]. In this section, we present dense reconstruction, trajectory estimation against ground truth, pose evolution, and the corresponding Absolute Pose Error (APE). Since we are interested in evaluating the overall trajectory drift, we mainly used the APE. The quantitative results are reported as the average of five runs, as the result is not exactly the same for each run. Due to the challenges associated with obtaining synchronized depth, RGB, and IMU datasets, we evaluated our algorithm on the TUM dataset without incorporating IMU data. This allowed us to utilize the available ground truth for trajectory evaluation. Since ground truth is not available for directly assessing the global dense map, we conducted a qualitative evaluation. However, by analyzing the trajectory errors compared to the ground truth, we can still infer the accuracy of the map to some extent.

For the *freiburg2_xyz* sequence, the dense map reconstruction result is shown in Figure 4.13b. The full estimated trajectory, alongside the ground truth in a 3D perspective, is illustrated in Figure 4.14. To provide a clearer comparison, the estimated and ground truth poses along the individual XYZ axes are plotted in Figure 4.16. The resulting Absolute Pose Error (APE) for this sequence has an average of 0.0049 m, while the Root Mean Square Error (RMSE) is 0.0057 m. Figure 4.15 graphically presents the error per frame, offering a frame-wise insight into the estimation accuracy, and the error along each individual axis is further detailed in Figure 4.17.

Table 4.1 presents a quantitative comparison of localization accuracy with other VSLAM methods that are implemented for dense mapping, including iMAP [64], NICE-SLAM [65], and ESLAM [66] to demonstrate the effectiveness of our proposed approach.



(a) RGB image environment for *freiburg2_xyz* dataset



(b) Reconstructed dense map for *freiburg2_xyz* dataset

Figure 4.13: RGB image based environment and corresponding reconstructed dense map for *freiburg2_xyz* dataset

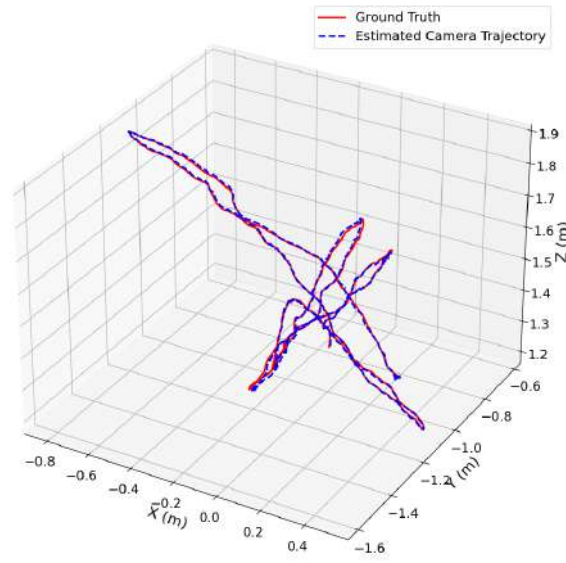


Figure 4.14: 3D trajectory and ground truth for the *freiburg2_xyz* sequence.

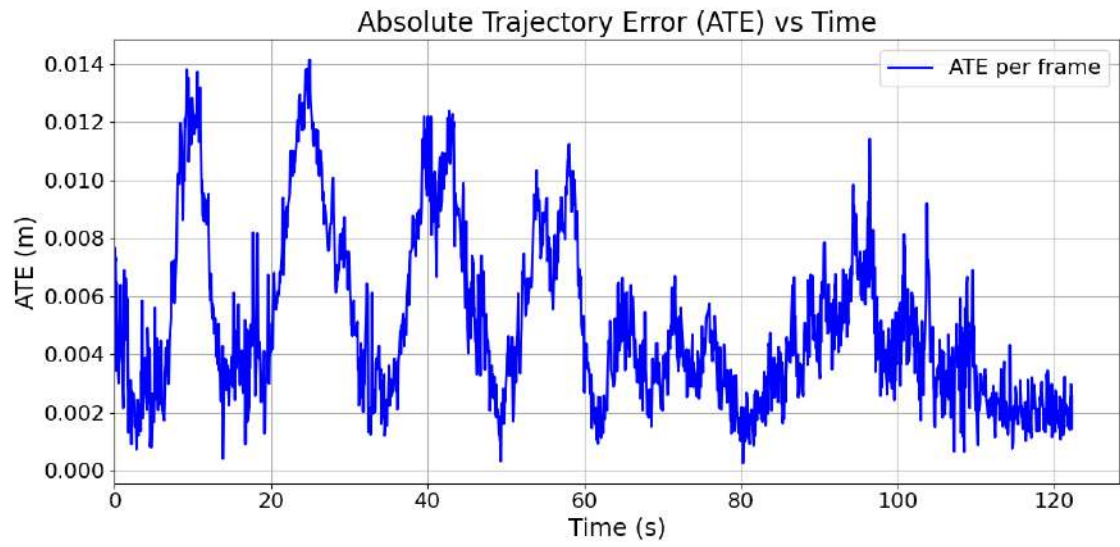


Figure 4.15: Absolute Trajectory Error (ATE) per frame on the *fr2_xyz* sequence.

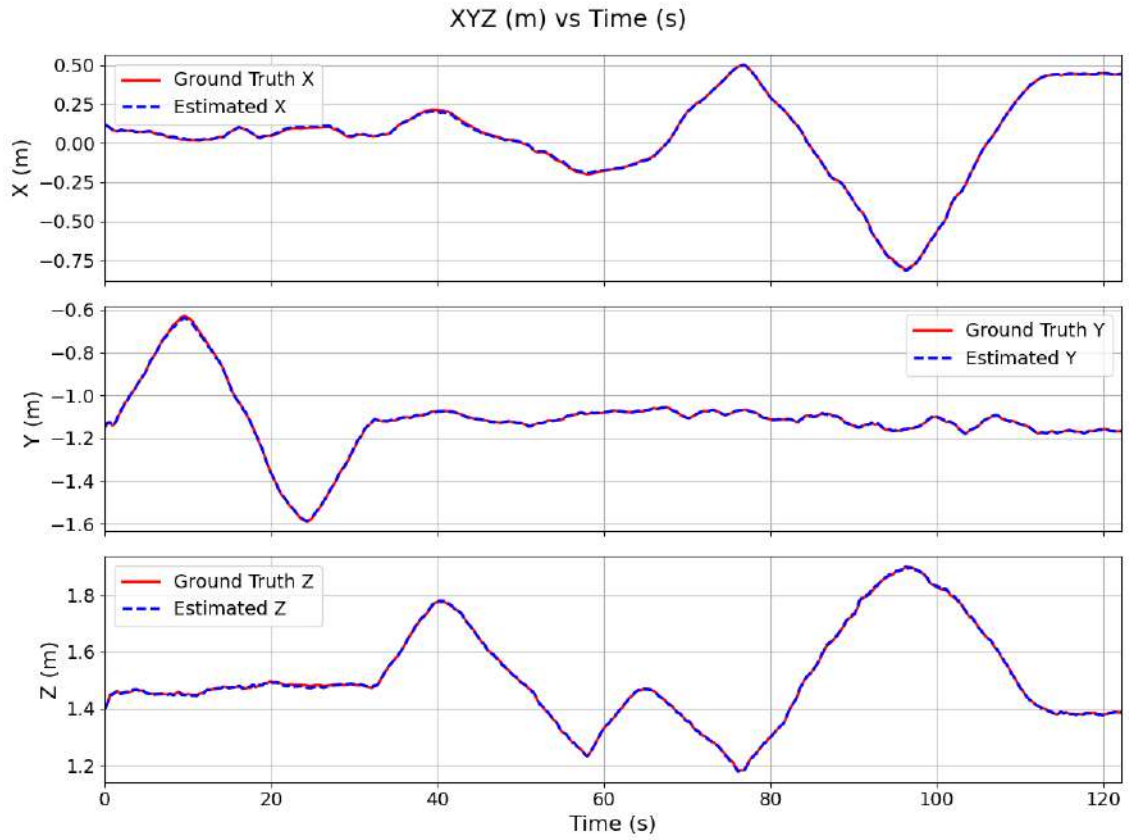


Figure 4.16: XYZ position components of the *freiburg2_xyz* sequence along with ground truth, used to evaluate the accuracy of the estimated trajectory at each time step.

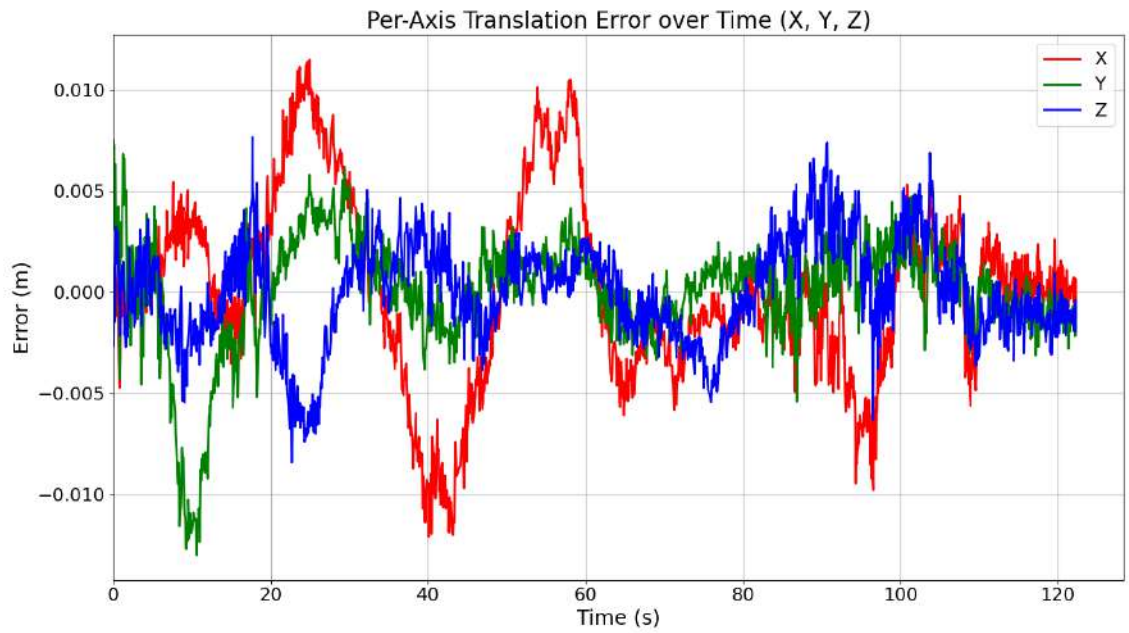


Figure 4.17: Per-axis XYZ error over time for the *freiburg2_xyz* sequence.

Table 4.1: RMSE comparison (in meters) on the *freiburg2_xyz* dataset using different dense VSLAM methods, the RMSE of iMAP [64], NICE-SLAM [65], and ESLAM [66] are adapted [66]. **Bold** achieves the lowest RMSE(higher accuracy).

Method	RMSE (m)
iMAP*	0.0205
NICE-SLAM	0.0239
ESLAM	0.0111
Proposed method(Ours)	0.0057

For the *freiburg3_long_office_household* sequence, the reconstructed dense map result is shown in Figure 4.19, and the RGB images of the environment from both viewpoints are presented in Figure 4.18. The estimated trajectory compared to the ground truth is shown in Figure 4.20 (3D view). To better understand the motion dynamics, the evolution of translational components (XYZ) is shown in Figure 4.22. The APE for this sequence has an average of 0.011 m and an RMSE of 0.0127 m. The corresponding APE plot per frame is shown in Figure 4.21, while Figure 4.23 shows the error along each axis over time.

Table 4.2 presents the RMSE comparison on the *freiburg3_office_household* sequence using various dense VSLAM methods, including iMAP [64], NICE-SLAM [65], and ESLAM [66]. Our method achieves the lowest RMSE, demonstrating significantly improved localization accuracy compared to existing approaches.



(a) Front view of the environment in the RGB image from the *freiburg3_office_household* sequence.



(b) Another RGB view from the *freiburg3_office_household* sequence.

Figure 4.18: RGB images from different viewpoints in the *freiburg3_office_household* sequence from the TUM RGB-D dataset.

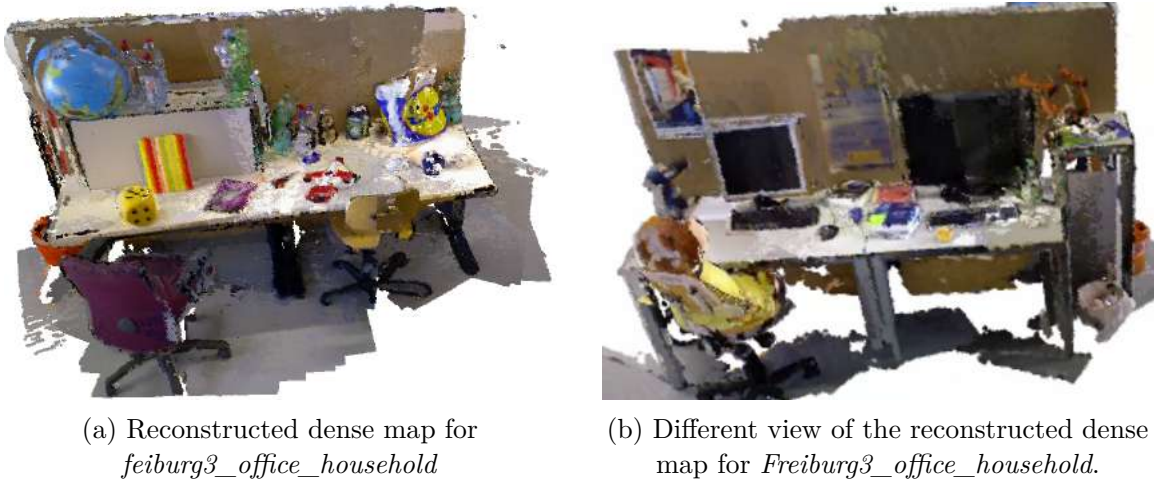


Figure 4.19: Reconstructed dense maps of the *freiburg3_office_household* sequence from different viewpoints from TUM RGB-D datasets.

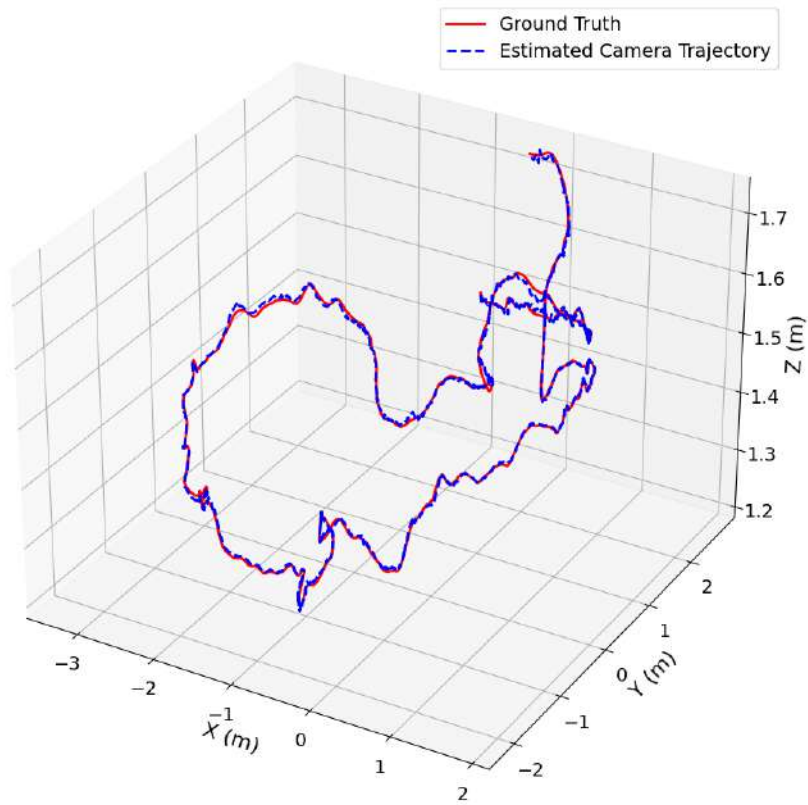


Figure 4.20: 3D (XYZ) estimated trajectory and ground truth for *Fr3_household*.

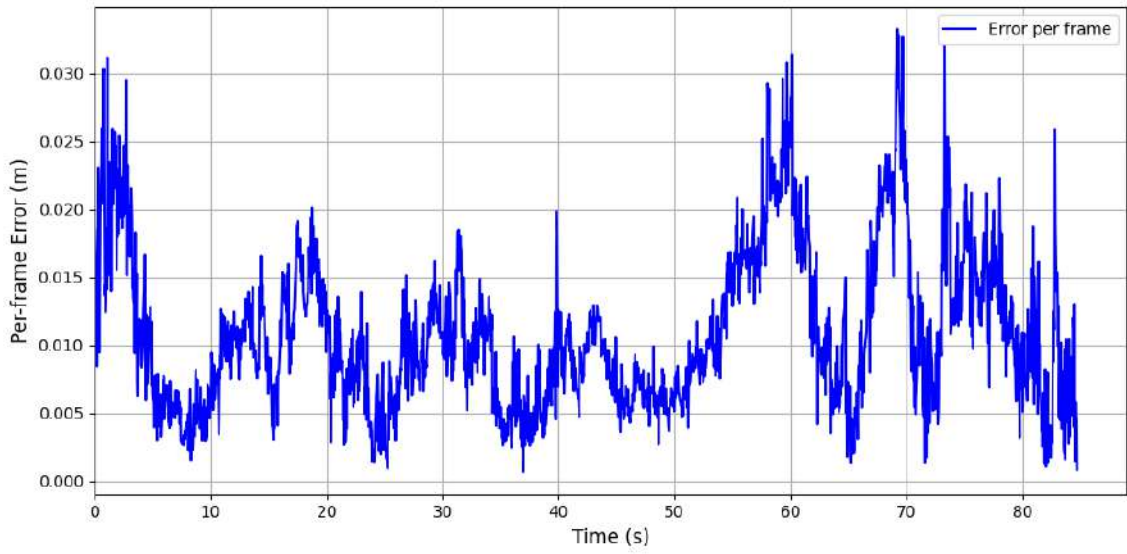


Figure 4.21: Per-frame error for *Fr3_household*.

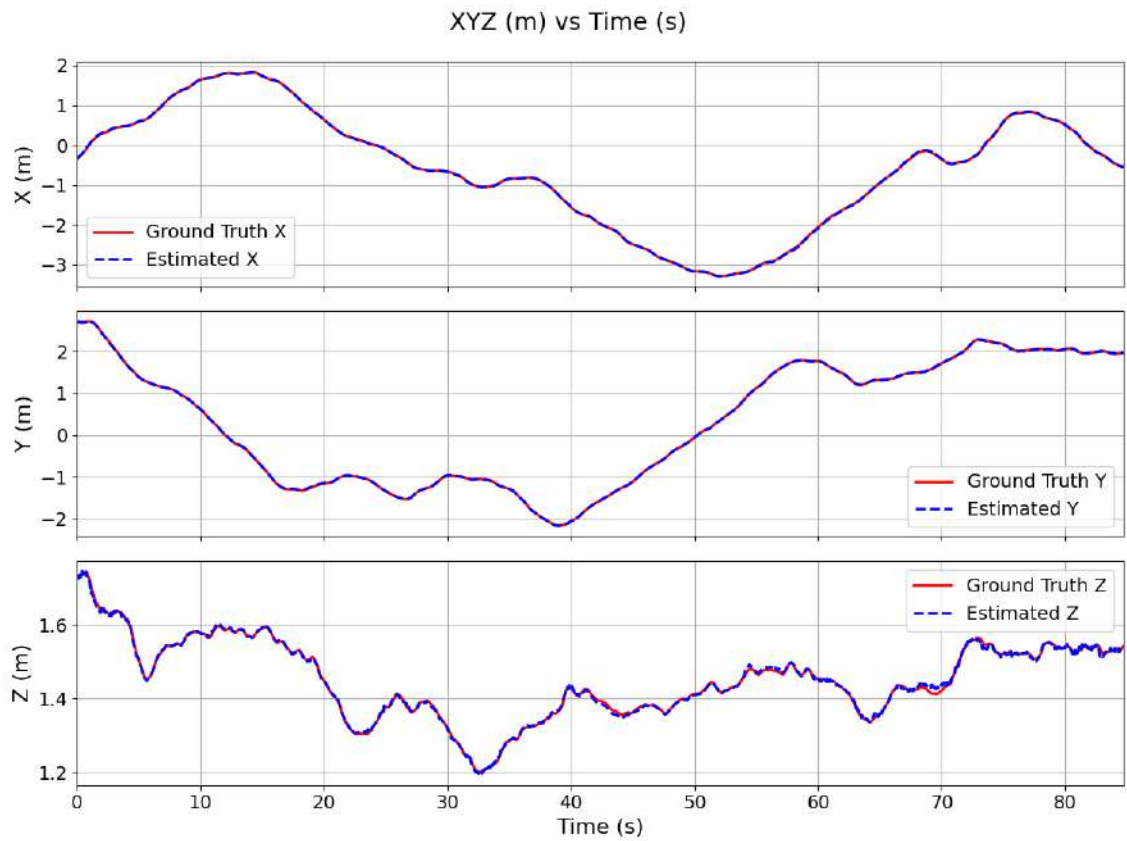


Figure 4.22: XYZ position components of the *freiburg3_office_household* sequence along with ground truth, used to evaluate the accuracy of the estimated trajectory at each time step.

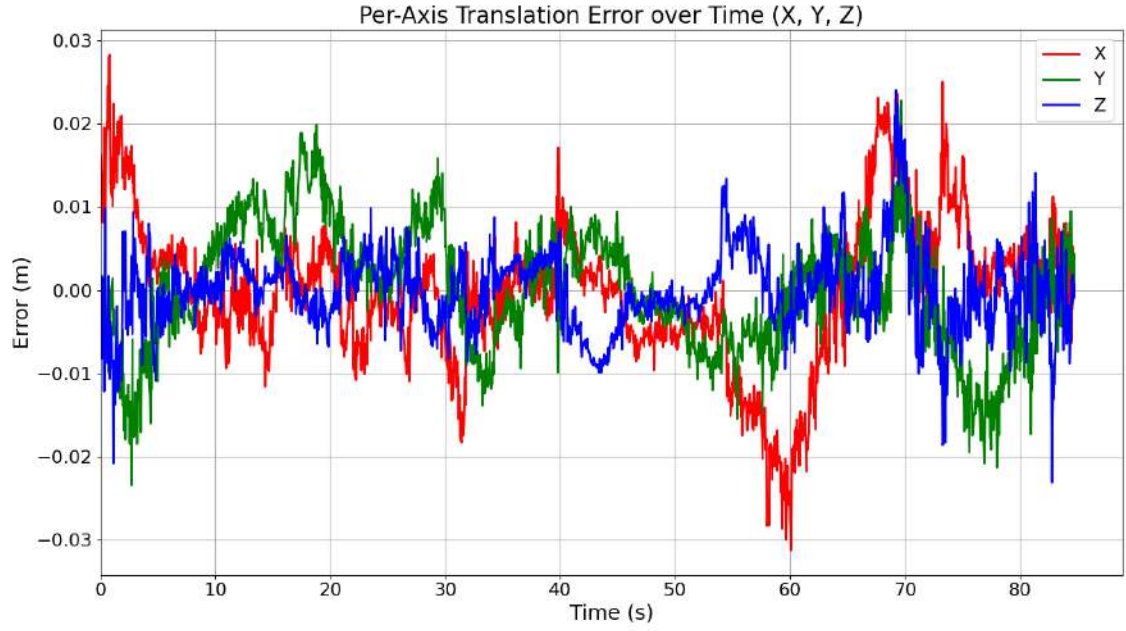


Figure 4.23: Per-axis XYZ error over time for the *freiburg3_long_house* sequence.

Table 4.2: RMSE comparison (in meters) on the *freiburg3_office_household* dataset using different dense VSLAM methods. The RMSE values are adapted from [66]. The proposed method (**bold**) achieves the lowest RMSE, indicating superior performance.

Method	RMSE (m)
iMAP*	0.058
NICE-SLAM	0.0302
ESLAM	0.0242
Proposed method(Ours)	0.0127

Although the estimated trajectory is aligned to the ground truth using the Umeyama method [67], the errors of the aligned trajectory do not start at zero. This is due to the algorithm of the Umeyama alignment, which computes a global similarity transformation, including rotation, translation, and uniform scaling that minimizes the total squared error across all points in the trajectory. As a result, the alignment does not guarantee that the first estimated pose exactly matches the first ground truth pose, nor does it ensure that the initial error is zero. The transformation is optimized to minimize the overall trajectory error, not the error at any specific point. Therefore, it is typical for the aligned trajectory to start near—but not exactly at—the ground truth start point, and for the per-frame Absolute Trajectory Error (ATE) to begin at a non-zero value.

Table 4.3 presents a summary of the evaluation results on selected sequences from the TUM RGB-D benchmark data sets. The table highlights the performance of the algorithm

in terms of APE, which includes the maximum and minimum values, and RMSE, providing information on its accuracy and robustness in various motion patterns and environmental conditions.

Table 4.3: Absolute Pose Error (APE) and RMSE for TUM RGB-D benchmark sequences, with translational (m) and rotational (deg) values.

trans—> *Translational* and *rot*—> *Rotational*

Sequence	Min		Max		Mean		RMSE	
	trans	rot	trans	rot	trans	rot	trans	rot
fr1_xyz	0.0005	0.6905	0.0385	4.1691	0.0110	2.2291	0.0125	2.3138
fr1_rpy	0.0018	6.3990	0.0873	11.1283	0.0209	8.2366	0.0256	8.2776
fr1_desk	0.0019	0.1746	0.0711	6.1074	0.0178	1.4731	0.0205	1.5839
fr2_xyz	0.0004	0.3192	0.0141	2.0648	0.0049	0.9680	0.0057	1.0012
fr2_desk	0.0006	0.0232	0.0986	2.7645	0.0118	0.4822	0.0135	0.5519
fr3_long office	0.0011	0.1561	0.0347	2.0486	0.0112	1.0121	0.0127	1.0483
fr3-sitting-static	0.0009	9.1457	0.0354	10.5117	0.0056	9.4957	0.0066	9.4968

To evaluate the accuracy of our trajectory estimation, we compared our results with several other state-of-the-art dense mapping methods by analyzing their reported RMSE values on multiple sequences from the TUM RGB-D dataset. In particular, we focused on the RMSE metric as a primary comparison of the performance. For RGB-D SLAM systems, we primarily referenced the benchmark presented in [68]. Additionally, we included comparisons with MRSMap [69] and graph-based visual SLAM approaches [70], among others. The RMSE value is adapted from their reported results for comparison with our method. These references provided a basis for assessing the relative accuracy of our approach.

Table 4.4: RMSE comparison of our method with state-of-the-art approaches on selected TUM RGB-D sequences. Lower RMSE values indicate better trajectory accuracy.

Missing entries (denoted by '-') indicate that the corresponding method was not evaluated on that sequence.

Sequence	Ours	RGB-D SLAM	MRSMap	Graph-VSLAM
fr1_xyz	0.0125	0.021	0.013	0.044
fr1_rpy	0.0256	0.042	0.027	0.056
fr1_desk	0.0205	0.049	0.043	0.089
fr2_xyz	0.0057	-	0.020	-
fr2_desk	0.0135	-	0.052	-
fr3_long office	0.0127	-	-	-
fr3_sitting_static	0.0066	-	-	-

Based on Table 4.4, the RMSE of our method is lower than that of the methods used for comparison, for the sequences we evaluated.

4.3 Limitations

As this work focuses on camera features based on SLAM techniques, it is therefore constrained by the limitations of vision-based systems. The evaluation is conducted predominantly in indoor and partially in outdoor environments, with limited analysis in highly dynamic or low-light scenarios. One significant limitation arises from the restricted field of view of the camera, which prevents comprehensive coverage of the entire environment and results in incomplete scene reconstruction. Additionally, the depth estimation is limited to a maximum range of approximately 20 meters, which significantly reduces the effectiveness of this method in large-scale or open outdoor environments. Moreover, the system does not explicitly handle dynamic elements in the scene; movable objects such as people or vehicles are not filtered out or segmented, which may adversely affect both localization and mapping accuracy. The method also assumes relatively static environments, and as such, its robustness in highly dynamic conditions remains unverified. Additionally, there are no available resources that use ROS 2-based dense mapping for visual-inertial SLAM, so we couldn't compare our results for both indoor and outdoor ROSBag-based data collections.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In conclusion, this thesis focuses on visual-inertial SLAM using camera and inertial sensor inputs. Cameras were selected for their lightweight and cost-effective nature, despite inherent limitations such as a narrow field of view and challenges in depth estimation, particularly in outdoor environments. A custom dataset in ROS2 bag was collected to evaluate dense map reconstruction by observing structural features like edges and corners in the absence of ground truth. Two mapping strategies were implemented: one that uses every frame with ICP registration based on camera pose estimates, and another using keyframes, which give more robust data sets in indoor, outdoor and benchmark data sets, offering improved computational efficiency.

The trajectory estimation was validated against the ground truth using public datasets, mainly the TUM RGB-D dataset, and compared with state-of-the-art dense methods VSLAM. Quantitative evaluation of trajectory accuracy and error metrics, along with qualitative evaluation of dense maps, demonstrated that the proposed system achieves competitive or superior performance on benchmark datasets, despite the lack of ground truth for dense reconstruction. The main work of this thesis is to answer the following research questions.

- How can the integration of visual sensory data and IMU measurements improve the accuracy and robustness of VSLAM in different environments, including featureless environments? The integration of IMU data into the visual input only (VSLAM) addresses the limitations of cameras, particularly in environments with low texture and poor lighting conditions. The IMU provides high-frequency measurements

of linear acceleration and angular velocity, which are used to estimate short-term motion, especially when visual tracking becomes unreliable or fails entirely. In our experiments conducted, primarily in indoor environments, we observed that visual tracking can fail in dark or structureless areas, even if only for very short periods. By tightly coupling the visual and inertial data, the system maintains accurate pose estimation during these challenging conditions. The IMU effectively bridges the gaps in visual data, allowing for more robust data, particularly in environments where visual sensors are insufficient or temporarily unavailable.

- How can loop closure detection improve map consistency in VSLAM?

Loop closure detection addresses the problem of accumulated drift in pose estimation. In our system, loop detection is implemented using the method proposed in [18], which utilizes the DBoW approach. This technique compares the current keyframe's visual descriptor with previously stored keyframes, excluding those that are directly connected to the current keyframe in the covisibility graph within the active map—i.e., the submap currently used for tracking. Once a candidate loop is detected using DBoW, a similarity transformation is estimated using RANSAC to robustly align the current view with the candidate loop. If validated, this transformation is used to perform pose graph optimization, which adjusts all related keyframes and poses to enforce global consistency.

Because the reconstructed dense maps are built incrementally based on these poses, correcting the trajectory directly improves the quality of the map. Loop closure ensures that previously mapped structures are accurately aligned, reduces duplicate features, and eliminates spatial distortions. This results in a more consistent and globally accurate dense map. As a result, both localization accuracy and map consistency are significantly improved.

- How can we reconstruct a dense map in VSLAM under different environmental scenarios?

This work focused on developing methods for generating accurate dense maps in various types of environments, including public indoor, outdoor, and available benchmark data sets. Dense mapping requires reliable depth estimation and precise pose tracking. In our system for the custom dataset collected, we use synchronized visual-inertial data to estimate camera motion and integrate depth information over time

to build a dense map to reconstruct. Visual sensors provide high-resolution texture and depth, while the IMU contributes stability and continuity, especially during rapid motion or in environments with a short period of time with poor visual features. By combining these data within a tightly coupled VSLAM framework and incorporating strategies such as keyframe selection, we reconstructed consistent and detailed maps in various scenarios, from indoor rooms with well-defined geometry to open or low-feature areas where traditional vision-only methods struggle.

5.2 Future Work

Future work will focus on the following directions:

- Using high-performance hardware, particularly GPUs, to accelerate processing and enhance the accuracy of dense VSLAM reconstruction in large-scale environments, allowing real-time performance.
- Enhance dense mapping by integrating additional sensors to improve depth estimation at longer ranges, particularly in outdoor environments, or by leveraging deep learning techniques for more accurate depth prediction and scene understanding, given that camera-based depth estimation is typically limited to a maximum range of 20 meters.
- Current work does not address dense map reconstruction for reflective surfaces such as white windows; therefore, exploring alternative techniques, ideally deep learning-based methods, to solve this challenge would be a valuable direction for future research.

Bibliography

- [1] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. In: *Proc. European Conference on Computer Vision*. 2014, pp. 834–849.
- [2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. “ORB-SLAM: A versatile and accurate monocular SLAM system”. In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163.
- [3] R. Mur-Artal and J. D. Tardós. “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras”. In: *IEEE Transactions on Robotics* 33.5 (Oct. 2017), pp. 1255–1262.
- [4] D. Scaramuzza and F. Fraundorfer. “Visual odometry: part I—the first 30 years and fundamentals”. In: *IEEE Robot. Autom. Mag.* 18.4 (2011), pp. 80–92.
- [5] J. Campbell et al. “A robust visual odometry and precipice detection system using consumer-grade monocular vision”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*. 2005, pp. 3421–3427.
- [6] L. Matthies. “Error modeling in stereo navigation”. In: *IEEE J. Robot. Autom.* 3 (1987), pp. 239–250.
- [7] A. Talukder et al. “Real-time detection of moving objects in a dynamic scene from moving robotic vehicles”. In: *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. 2003.
- [8] C. Dornhege and A. Kleiner. “Visual odometry for tracked vehicles”. In: *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR)*. 2006.
- [9] C. Schmid, R. Mohr, and C. Bauckhage. “Evaluation of Interest Point Detectors”. In: *International Journal of Computer Vision* 37.2 (2000), pp. 151–172.
- [10] Yanlin Jin et al. “ORB-SfMLearner: ORB-Guided Self-supervised Visual Odometry with Selective Online Adaptation”. In: (2023).

- [11] MathWorks. *Monocular Visual-Inertial SLAM*. <https://www.mathworks.com/help/vision/ug/monocular-visual-inertial-slam.html>. Accessed: 2025-05-01. 2025.
- [12] A. J. Davison. *Real-Time Simultaneous Localisation and Mapping with a Single Camera*. Robotics Research Group, Dept. of Engineering Science, University of Oxford, UK. [Online]. Available: <http://www.robots.ox.ac.uk/~ajd/>. 2003.
- [13] A. J. Davison et al. “MonoSLAM: Real-time Single Camera SLAM”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (Aug. 2007), pp. 1052–1067.
- [14] D. Gálvez-López and J. D. Tardós. “Bags of binary words for fast place recognition in image sequences”. In: *IEEE Transactions on Robotics* 28.5 (Oct. 2012), pp. 1188–1197.
- [15] C. Forster, M. Pizzoli, and D. Scaramuzza. “SVO: Fast semi-direct monocular visual odometry”. In: *Proc. IEEE Int. Conf. Robotics and Automation*. 2014, pp. 15–22.
- [16] C. Forster et al. “SVO: Semidirect visual odometry for monocular and multicamera systems”. In: *IEEE Transactions on Robotics* 33.2 (Apr. 2017), pp. 249–265.
- [17] R. Mur-Artal and J. D. Tardós. “Visual-inertial monocular SLAM with map reuse”. In: *IEEE Robotics and Automation Letters* 2.2 (Apr. 2017), pp. 796–803.
- [18] Juan J. Gómez Rodríguez Carlos Campos Richard Elvira. *"An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM"*. 2021.
- [19] A. Martinelli. “Closed-form solution of visual-inertial structure from motion”. In: *International Journal of Computer Vision* 106.2 (2014), pp. 138–152.
- [20] J. Kaiser et al. “Simultaneous state initialization and gyroscope bias calibration in visual-inertial aided navigation”. In: *IEEE Robotics and Automation Letters* 2.1 (Jan. 2017), pp. 18–25.
- [21] T. Qin, P. Li, and S. Shen. “VINS-Mono: A robust and versatile monocular visual-inertial state estimator”. In: *IEEE Trans. Robot.* 34.4 (Aug. 2018), pp. 1004–1020.
- [22] T. Qin et al. *VINS-Fusion: An optimization-based multi-sensor state estimator*. 2019.

- [23] Z. Yang and S. Shen. “Monocular visual-inertial state estimation with online initialization and camera-IMU extrinsic calibration”. In: *IEEE Trans. Autom. Sci. Eng.* 14.1 (Jan. 2017), pp. 39–51.
- [24] S. Shen et al. “Initialization-free monocular visual-inertial estimation with application to autonomous MAVs”. In: *Proc. Int. Sym. Exp. Robot.* Marrakech, Morocco, June 2014, pp. 211–227.
- [25] T. Qin, P. Li, and S. Shen. *Relocalization, Global Optimization and Map Merging for Monocular Visual-Inertial SLAM*. arXiv preprint arXiv:1803.01549. 2018.
- [26] T. Lupton and S. Sukkarieh. “Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions”. In: *IEEE Transactions on Robotics* 28.1 (Feb. 2012), pp. 18–25.
- [27] P. Henry et al. “RGB-D Mapping: Using Depth Cameras for 3D Modeling of Indoor Environments”. In: 2010.
- [28] D. G. Lowe. “Distinctive image features from scale-invariant keypoints”. In: (2004).
- [29] R.A. Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. Basel, Switzerland, Oct. 2011, pp. 127–136.
- [30] Paul J. Besl and Neil D. McKay. “A Method for Registration of 3-D Shapes”. In: *Proceedings of the IEEE Conference on Pattern Analysis and Machine Intelligence*. IEEE, 1992.
- [31] Liang Cheng et al. “Registration of Laser Scanning Point Clouds: A Review”. In: *Sensors* 18.5 (2018), p. 1641.
- [32] Zhenpei Yang et al. “Extreme Relative Pose Estimation for RGBD Scans via Scene Completion”. In: *IEEE Access* 8 (2020), pp. 7362–7372.
- [33] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. “Feature-Metric Registration: A Fast Semi-Supervised Approach for Robust Point Cloud Registration Without Correspondences”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2020, pp. 11366–11374.
- [34] OpenCV Contributors. *Camera Calibration with OpenCV*. Online. Accessed: 2025-03-08. 2024. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.

- [35] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. second. Cambridge University Press, 2004.
- [36] C. Li. *Stereo Visual-Inertial Odometry*. <https://www.chunshangli.com/2018/12/20/stereo-vio.html>. 2018.
- [37] O. Guclu and A.B. Can. “k-SLAM: A Fast RGB-D SLAM Approach for Large Indoor Environments”. In: *Computer Vision and Image Understanding* (2019).
- [38] H. Bay et al. “Speeded-Up Robust Features (SURF)”. In: *Computer Vision and Image Understanding* 110 (2008), pp. 346–359.
- [39] E. Rublee et al. “ORB: An Efficient Alternative to SIFT or SURF”. In: *Proceedings of the 2011 International Conference on Computer Vision*. Barcelona, Spain, pp. 2564–2571.
- [40] A.M. Ali and M.J. Nordin. “SIFT-Based Monocular SLAM with Multi-Clouds Features for Indoor Navigation”. In: *Proceedings of the TENCON 2010—2010 IEEE Region 10 Conference*. 2010.
- [41] Xiang Dong et al. “FSD-SLAM: A Fast Semi-Direct SLAM Algorithm”. In: *Complex Intelligent Systems* 8 (2021). DOI: 10.1007/s40747-021-00323-y.
- [42] Ezio Malis Geraldo Silveira and Patrick Rives. “An Efficient Direct Approach to Visual SLAM”. In: *IEEE TRANSACTIONS ON ROBOTICS*. 2008.
- [43] X. He et al. “LiDAR-Visual-Inertial Odometry Based on Optimized Visual Point-Line Features”. In: *Remote Sensing*. 2022.
- [44] R.A. Newcombe, S.J. Lovegrove, and A.J. Davison. “DTAM: Dense Tracking and Mapping in Real-Time”. In: *Proceedings of the 2011 International Conference on Computer Vision*. 2011.
- [45] Weifeng Chen et al. “An Overview on Visual SLAM: From Tradition to Semantic”. In: *Remote Sensing*.
- [46] Open Robotics. “Robot Operating System 2 (ROS2) Humble”. In: *Open Robotics* (). URL: <https://docs.ros.org/en/humble/index.html>.
- [47] Stereolabs. “ZED Stereo Camera”. In: *Stereolabs* (). URL: <https://www.stereolabs.com/>.
- [48] OpenCV Documentation. “Optical Flow Tutorial”. In: *OpenCV* (). URL: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html.

- [49] Qiang Li et al. “An Improved Feature Matching ORB-SLAM Algorithm”. In: *Journal of Physics: Conference Series* 1693 (2020), p. 012068. DOI: 10.1088/1742-6596/1693/1/012068.
- [50] Joern Rehder et al. “Extending Kalibr: Calibrating the Extrinsic of Multiple IMUs and of Individual Axes”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 4304–4311.
- [51] Paul Furgale, Joern Rehder, and Roland Siegwart. “Unified Temporal and Spatial Calibration for Multi-Sensor Systems”. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, 2013.
- [52] Paul Furgale, T D Barfoot, and G Sibley. “Continuous-Time Batch Estimation Using Temporal Basis Functions”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. St. Paul, MN, 2012, pp. 2088–2095.
- [53] J. Maye, P. Furgale, and R. Siegwart. “Self-supervised Calibration for Robotic Systems”. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IVS)*. 2013.
- [54] L. Oth et al. “Rolling Shutter Camera Calibration”. In: *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*. 2013.
- [55] Shaowu Yang, Sebastian Scherer, and Andreas Zell. “Robust Onboard visual SLAM for autonomous MAVs”. In: vol. 302. July 2014. ISBN: 978-3-319-08337-7. DOI: 10.1007/978-3-319-08338-4_27.
- [56] Xiaoshui Huang, Guofeng Mei, Jian Zhang, Rana Abbas. *A comprehensive survey on point cloud registration*. <https://arxiv.org/>. 2021. URL: <https://arxiv.org/pdf/2103.02690>.
- [57] Paul J. Besl and Neil D. McKay. “A Method for Registration of 3-D Shapes”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256.
- [58] Seth D. Billings, Emad M. Boctor, and Russell H. Taylor. “Iterative Most-Likely Point Registration (IMLP): A Robust Algorithm for Computing Optimal Shape Alignment”. In: *PloS One* 10.3 (2015).

- [59] E. Rublee et al. “ORB: An Efficient Alternative to SIFT or SURF”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.
- [60] J. Sivic and A. Zisserman. “Video Google: A Text Retrieval Approach to Object Matching in Videos”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Vol. 3. IEEE Computer Society, 2003, pp. 1470–1470.
- [61] Soonhac Hong and Cang Ye. “Pose Graph Based Visual SLAM Algorithm for Robot Pose Estimation”. In: *Proceedings of the IEEE Conference*. IEEE.
- [62] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM*. <https://github.com/MichaelGrupp/evo>. 2017.
- [63] J. Sturm et al. “A Benchmark for the Evaluation of RGB-D SLAM Systems”. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.
- [64] Edgar Sucar et al. “imap: Implicit mapping and positioning in real-time”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 6229–6238.
- [65] Zihan Zhu et al. “Nice-slam: Neural implicit scalable encoding for slam”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 12786–12796.
- [66] Mohammad Mahdi Johari, Camilla Carta, and François Fleuret. *ESLAM: Efficient Dense SLAM System Based on Hybrid Representation of Signed Distance Fields*. 2023. arXiv: 2211.11704 [cs.CV]. URL: <https://arxiv.org/abs/2211.11704>.
- [67] Shinji Umeyama. “Least-squares estimation of transformation parameters between two point patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380.
- [68] Felix Endres et al. “An evaluation of the RGB-D SLAM system”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 1691–1696. DOI: 10.1109/ICRA.2012.6225199.
- [69] Jörg Stückler and Sven Behnke. “Integrating depth and color cues for dense multi-resolution scene mapping using RGB-D cameras”. In: *2012 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. 2012, pp. 162–167. DOI: 10.1109/MFI.2012.6343050.

- [70] Jan Helge Klüssendorff et al. “Graph-based visual SLAM and visual odometry using an RGB-D camera”. In: *9th International Workshop on Robot Motion and Control*. 2013, pp. 288–293. DOI: 10.1109/RoMoCo.2013.6614623.

5

LiDAR Odometry and Mapping Beyond RTK Accuracy
Liviu-Daniel Florescu



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTER IN
INTELLIGENT FIELD ROBOTIC SYSTEMS

LiDAR Odometry and Mapping Beyond RTK Accuracy

Author:

Liviu-Daniel Florescu

MSc Intelligent Field Robotic Systems (IFRoS)

Internal supervisor:

Iván Eichhardt

Assistant Professor

External supervisor:

Maximilian Fenkart

CTO, Sodex Innovations GmbH

Budapest, 2025

Title of the thesis

LiDAR Odometry and Mapping Beyond RTK Accuracy

Topic of the thesis

Out of the five basic senses that humans use to experience the world, vision accounts for 80% of the information input that our brains operate with [1]. Naturally, robotics research aims to replicate this and develop systems that can not only collect high-quality visual data but also create rich artificial representations of the world, enabling autonomous systems to confidently reason about their environment.

This work will focus on the use of 3D Light Detection and Ranging (LiDAR) sensors in outdoor settings. When mounted on an arbitrary mobile base and moved around a target environment, the sensor collects information about the geometry of the scene, which can be merged in order to create a general 3D model. However, this process depends on accurate displacement measurements that are not trivial to obtain. An existing solution relies on Global Navigation Satellite System (GNSS) localization, corrected using Real-Time Kinematic positioning (RTK), and orientation from an Inertial Measurement Unit (IMU). Together, these create an Inertial Navigation System (INS) whose output can be interpreted as the 3D transformation between sensor poses at discrete time steps. Even though this represents the state-of-the-art technology for outdoor localization, with centimeter-level position error, its accuracy is unsatisfactory when it comes to high-quality pointcloud registration. Additionally, this is not feasible for all outdoor scenes, because GNSS accuracy varies heavily depending on surroundings and signal strength.

We will investigate methods that address the limitations of the INS-based registration, by using the visual information in the scene, such that the system is less reliant on a sensor with fluctuating uncertainty. Previous research in this area [2] indicates that visual cues alone should be enough to achieve reliable displacement estimation, enabling the computation of odometry from LiDAR data, as well as creating a 3D map of the explored environment. A comparison between LiDAR odometry and GNSS localization is also within the scope of this work. The research questions that we aim to answer are the following:

- What metrics exist for measuring the accuracy of pointcloud registration?
- Can methods that use only visual information achieve higher quality pointcloud registration (3D mapping) than RTK-based merging?
- To what extent is LiDAR-based odometry an alternative to GNSS localization?

The work will be carried out in collaboration with Sodex Innovations GmbH, who provide the sensor rig (LiDAR, RGB cameras, GNSS + RTK + IMU) as well as several existing datasets consisting of sensor data collected while exploring outdoor rural environments. The project will span approximately 15 weeks, and is tentatively structured as shown below, subject to change brought by results at different stages.

Stage	Estimated Duration	Description
Introduction	2 weeks	Familiarization with the existing sensors, datasets and related processes
Background investigation	3 weeks	Exploring existing solutions, initial experimentation on available data
Design and Development	4 weeks	Iterative implementation and testing of new approaches
Analysis	4 weeks	Evaluation and comparison between the solutions developed at different iterations
Final write-up	2 weeks	Reporting methods, implementation and results

Keywords

LiDAR odometry, LiDAR mapping, pointcloud registration, GNSS, RTK

References

- [1] Man, Dariusz & Olchawa, Ryszard. (2018). The Possibilities of Using BCI Technology in Biomedical Engineering. 10.1007/978-3-319-75025-5_4.
- [2] Lee, Dongjae & Jung, Minwoo & Yang, Wooseong & Kim, Ayoung. (2024). LiDAR odometry survey: recent advancements and remaining challenges. Intelligent Service Robotics. 17. 1-24. 10.1007/s11370-024-00515-8.

Acknowledgements

This work, as well as my fruitful IFRoS journey, would not have been possible without the direct or indirect contributions of several individuals, whom I would like to express my gratitude towards.

To Iván Eichhardt, my academic supervisor, for his availability, detailed feedback, and guidance throughout the final term.

To Max Fenkart, my industrial supervisor, for his enthusiasm, ambitious ideas, and patience with my scientific approach.

To Imma Guell, the IFRoS professors at Universitat de Girona, and Dr. Zoltán Istenes, for their initiatives and restless dedication to this programme.

To my fellow international colleagues, for their humor, peer support, perseverance and contagious eagerness to explore.

To Dr. Michael Tandy and my previous colleagues at Ocado Technology, for encouraging me to pursue this Master's Degree.

To my family, for their continued help and logistical efforts to make this process as smooth as possible.

To my fiancée, for her daily emotional support, inspirational resilience, heartfelt care, and being an unwavering source of motivation.

To God, my faithful *refuge and strength, a very present help in trouble*.¹

¹Psalm 46:1, New King James Version

Contents

Acknowledgements

1	Introduction	3
1.1	Robotic perception	4
1.2	Problem definition	5
2	Background and Literature Review	7
2.1	Simultaneous Localization and Mapping (SLAM)	7
2.2	Point cloud Registration	8
2.3	LiDAR-based Odometry and Mapping	9
3	Methodology	12
3.1	Prerequisites	12
3.2	Hardware	13
3.2.1	LiDAR Sensor	14
3.2.2	GNSS/INS receiver	15
3.3	Data acquisition and pre-processing	15
3.3.1	Motion compensation	17
3.4	Solution architecture	19
3.4.1	Overview	20
3.4.2	Reconciling motion prediction, GNSS and registration	21
3.4.3	Motion prediction	24
3.4.4	Point Cloud Registration	27
3.4.5	Map Handling	32
3.4.6	Graph Optimization	33
3.5	Implementation details	35
4	Results	36
4.1	Evaluation metrics	36

CONTENTS

4.2	Parameter analysis	38
4.2.1	Point cloud voxelization	38
4.2.2	Local map size	39
4.2.3	Registration strategy	40
4.3	Odometry evaluation	41
4.4	GPS-integrated evaluation	43
4.5	Mapping evaluation	46
5	Conclusion	49
5.1	Future Work	51
	Bibliography	52
	List of Figures	62
	Acronyms	63
	Glossary	64

Chapter 1

Introduction

Like many scientific terms that we encounter in our daily activities, *robotics* describes a broad collection of technologies and stands at the intersection of key research directions. Motivated by practicality, disciplines that appear completely unrelated find themselves building upon advancements in other fields, diluting boundaries and joining forces to enable otherwise-impossible advancements.

What constitutes a *robot*, then? In its inceptive use by Karel Čapek [1] in 1921, the word stemmed from the Slavic *robota*, meaning “servitude” or “forced work”, and referred to a human-like mechanical system working on factory assembly lines. This concept, however, dates from earlier centuries. Around 1495, Leonardo Da Vinci envisioned a mechanical knight controlled by a series of pulleys, that was able to perform simple movements [2]. Testifying to the industrial revolution and the computational breakthroughs of the last few decades, modern-day humanoid robots can perform acrobatics [3], interact with humans in constrained scenarios [4, 5, 6] and even replicate human facial expressions [7]. Still, a device ought not necessarily appear human-like in order for it to be labeled as a robot. Autonomous vacuum cleaners, space rovers or crop-monitoring drones fall under the same category. At this stage, a complete taxonomy would have to address dozens of physical (size, shape, mobility, locomotion system, etc.) and non-physical (autonomy, perception abilities, use-case, etc.) characteristics, and none of these would independently convey the meaning that we intuitively associate to the notion of *robot*. Without assessing whether an exhaustive, generic definition is even possible, we can synthesize the above by affirming that a *robot* is an artificial system that performs one or more tasks and is able to evaluate its state or gather information from its environment.

1.1 Robotic perception

This formulation highlights two essential aspects of a robotic agent (Fig. 1.1): actuation, seen as some form of dynamic physical ability, allowing the agent to enact the desired behavior, and perception, the ability to observe changes in the environment.

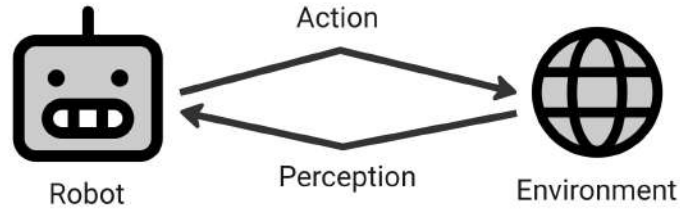


Figure 1.1: A simplified interpretation of the robotics paradigm. The interaction between an agent and its environment can be seen as a two-way flow: the agent alters the environment through its actions, and uses perception to observe it.

Sensing modalities have largely different contributions to the perception mechanism. To this extent, the phrase *visual dominance* was introduced by F. Colavita [8], whose study demonstrated that humans focus more on the visual component when presented with an audiovisual stimulus, and following research has strongly confirmed this tendency [9, 10]. Unsurprisingly, a similar pattern is emerging in the case of robots, thanks to the reduced cost, familiarity and wide availability of cameras. In many situations, visual stimuli provide most of the necessary information, and this has motivated the development of various image processing algorithms.

Technological innovation in the last century has led to the situation in which artificial sensors surpass humans in both the range of signals that are perceived, as well as the accuracy of the measurements. A relevant example is the class of Light Detection and Ranging (LiDAR) sensors which retrieve three-dimensional information about the environment at a very high frequency and with rather negligible measurement errors, in the form of *point clouds* (Fig. 1.2). Among many applications, this type of sensors can be used to construct virtual representations of a specific environment, enabling engineers to experiment with a realistic model, evaluate construction progress or validate a finished project.

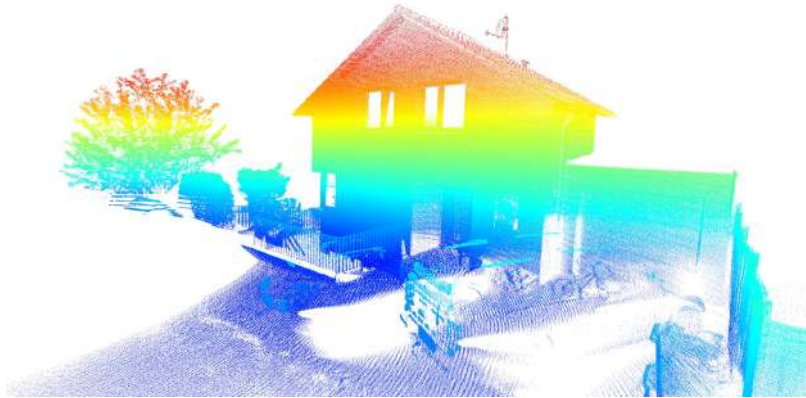


Figure 1.2: An example point cloud (points are colored by height), capturing the 3D representation of an outdoor scene. Multiple individual scans are placed into the same coordinate system to obtain a dense spatial representation.

1.2 Problem definition

The current work addresses a common and well-known problem in the area of Field Robotics, namely Simultaneous Localization and Mapping (SLAM), and combines the practicality of an industrial solution with a perception-based approach.

SDX-Compact (Fig. 3.1) is the main product of Sodex Innovations GmbH², consisting of multiple sensors that collect spatial and visual data. This module can be easily mounted on an arbitrary vehicle in order to expand its perception capabilities and convert it into a surveying device. While the vehicle is moving, the LiDAR sensor captures 3D scans of the surroundings, as well as related metadata (timestamps, localization information, signal intensity etc.). Because the rig includes a high-accuracy Inertial Navigation System (INS), the localization and orientation data can be used to join the collected point clouds and create a global 3D map of the traversed space.

Nonetheless, this suffers from two main limitations:

- Reliance on unstable signal: internally, the INS depends on information from a Global Navigation Satellite System (GNSS) receiver, which is limited to outdoor spaces and whose availability varies depending on weather conditions and surroundings (e.g. thick vegetation, tall buildings, bridges).
- Unsatisfactory accuracy: when merging point cloud data, positioning or localization errors introduce inconsistencies in the final 3D model, which hinders precise planning and construction.

²<https://www.sodex-innovations.com/>

1. Introduction

Our work aims to address these limitations by introducing a component that utilizes the information collected by the LiDAR sensor in conjunction with the existing data. Three research questions have been formulated to guide this process:

1. What metrics exist for measuring the accuracy of point cloud registration? In this context, registration refers to placing a pair of related point sets in a common reference frame.
2. Can methods that use only visual information achieve higher quality point cloud registration (3D mapping) than merging based on Real-Time Kinematics (RTK)? Usually, GNSS systems provide meter-level accuracy. In the current scenario, however, the system is corrected using Real-Time Kinematics, such that the expected error is at centimeter-level.
3. To what extent is LiDAR-based odometry an alternative to GNSS localization? Previous research indicates that the spatial information present in 3D point clouds could be used to compute the relative displacement between consecutive scans, resulting in the ability to estimate odometry (an essential component of robotic localization) without dedicated sensors such as wheel encoders or accelerometers.

The main contribution of the project consists of developing an original framework for localization and mapping based on data collected with an industrial sensor rig. The results are more generic than if a particular physical robotic system were involved, and thus are relevant for virtually any robotic application with a similar setup.

The following chapters of this document will cover related research directions and efforts that our work builds upon (Chapter 2), a detailed description of the components and algorithms involved in developing the project (Chapter 3), an evaluation of the method based on its results (Chapter 4), as well as a series of conclusions that were drawn from the overall process (Chapter 5).

Chapter 2

Background and Literature Review

The goal of this chapter is to present the research context in which our project was conducted, by looking at common methods for each of the main components and highlighting those that inspired the current approach.

2.1 Simultaneous Localization and Mapping (SLAM)

SLAM constitutes a key research area in robotics, because it is a foundational building block for autonomous operation. This problem occurs when the robot does not have prior access to a map of the environment, so it must construct one while keeping track of its current position (*online* SLAM). If the goal is to optimize the entire sequence of poses along the robot path, this is known as the *full* SLAM problem. [11]

Usually, the problem is discretized along the time dimension, such that at time t we aim to compute the posterior function $p(x_t, M | z_{1:t}, u_{1:t})$, where x_t is the current state, M is the map, $z_{1:t}$ represents the set of measurements collected so far, and $u_{1:t}$ the control sequence. As each of these variables can have different concrete representations, depending on the task and the available sensors, a broad range of approaches have been proposed.

Aulinas et al. [12] note that the earliest methods rely on probabilistic models derived from the recursive Bayes rule, as such formulations can provide intuitive representations of the various noisy components involved in a robotic system. When employing the Kalman Filter (KF) [13] or its variations, the robot state, measurements and control inputs are modeled as multi-dimensional Gaussians, whose covariances describe the associated uncertainty. The algorithm alternates between *predictions*, when the state is modified based on u_t , and *updates*, when the prediction is evaluated against the latest observation z_t , and the

2. Background and Literature Review

state hypothesis is updated accordingly. The Extended Kalman Filter (EKF) maintains this structure but can accommodate non-linearities in the displacement or measurement functions by using local linear approximations. Such methods have been successfully applied for indoor [14], aerial [15], and underwater [16] robots.

Particle Filter (PF), introduced by Del Moral [17], is a probabilistic approach that relies on the Monte Carlo method. Instead of an analytical form, the uncertainty is accounted for by considering a large set of samples (representing potential states) and weighting them based on measurement likelihood. This has a higher computational cost than the standard KF methods, but is not affected by any linearization limitations. Nie et al. [18] implemented a LiDAR SLAM algorithm based on PF localization, albeit for 2D mapping.

Another large group of methods is represented by *Visual SLAM*, when cameras are the main (and sometimes only) sensor used. As early as 1980, Moravec [19] developed a robot capable of estimating its motion by matching features in images captured at discrete poses, in a stop-and-go fashion. In 2007, SLAM was being performed using a single handheld camera [20, 21], triggering a separation from the popular, offline, Structure from Motion (SfM) techniques. The solution is even more robust when a stereo system is available [22], as this helps avoid the geometrical limitations of monocular vision.

2.2 Point cloud Registration

Before discussing LiDAR-based approaches in more detail, let us review the task of point cloud registration. Given two sets of points P, Q in arbitrary reference frames, representing (at least partially) the same scene, the goal is to find the transformation $\mathbf{T} \in \text{SE}(3)$ (translation and rotation) that best aligns the points. This can be expressed as an optimization problem:

$$\underset{\mathbf{T}}{\operatorname{argmin}} J(\mathbf{T}P, Q), \quad (2.1)$$

where J is a custom cost function. The particular case where known point correspondences are provided has a closed-form solution [23], but the Iterative Closest Point (ICP) algorithm [24] removes this constraint by alternating between correspondence generation — each point in P is paired with its nearest neighbor from Q — and minimizing the point-pair distances. This approach is by far the most widespread, thanks to its simplic-

2. Background and Literature Review

ity, computational efficiency (e.g., using a k-d tree for closest-point search) and many available variations [25, 26].

Chen and Medioni [27] proposed a “point-to-plane” minimization, while Segal et al. [28] formulated a generalized cost function by considering the probabilistic model underlying the point sets, and implemented a “plane-to-plane” minimization. Other variants modify the point selection strategy [29, 30], apply a weight to each pair [31], prune the set of the correspondences [32, 33] or employ a different minimization algorithm [34].

A different class of solutions operates by discretizing the space into voxels and estimating a normal distribution using the points in each voxel. Introduced in 2003, the Normal Distributions Transform (NDT) [35] algorithm is the main competitor to ICP, enabling registration to be performed without the need for point-to-point correspondences. Magnusson et al. [36] extended this work to 3D scans, and performed a thorough comparison between this and ICP [37] for the challenging task of registering point clouds captured in underground mines, where few usable features are present. More recently, the method has been applied for outdoor mapping [38], and as the basis for a Lidar Odometry and Mapping solution [39].

Neural Networks can also be used for point cloud registration, by creating correspondences based on features extracted by a descriptor architecture [40, 41]. This is particularly useful when no initial estimate of the transformation between the two point sets is available — a scenario that the other approaches cannot directly handle.

2.3 LiDAR-based Odometry and Mapping

Despite their high price range, LiDAR sensors have gained significant popularity in the last decade, becoming the go-to solution for autonomous mobility. Unlike cameras, LiDARs are not dependent on ambient lighting, so they can operate in total darkness, and they provide high-frequency, reliable 3D information without the need for additional processing. Basic applications include obstacle detection [42, 43], but the amount of information they provide makes them excellent for localization and mapping purposes.

A key reference is LiDAR Odometry and Mapping (LOAM), the work of Zhang and Singh [44], who used a motorized, 2-axis LiDAR scanner to generate 3D point clouds. To register consecutive scans, they perform feature extraction (sharp edges or planar patches), identify correspondences to the previous scan, and find the optimal transformation using the Levenberg-Marquardt method [60, 61]. To account for displacement during the beam

2. Background and Literature Review

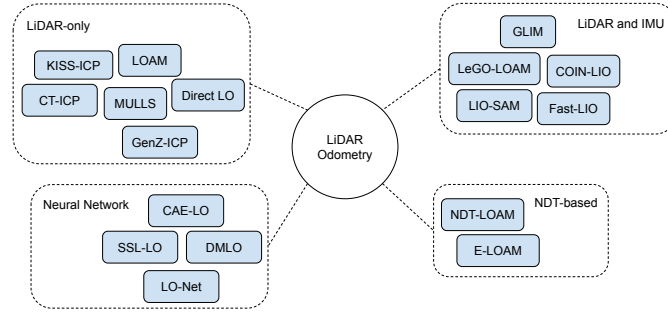


Figure 2.1: A visualization of LiDAR odometry methods, grouped by approach: LOAM [44], KISS-ICP [45], CT-ICP [46], MULLS [47], Direct LO [48], GenZ-ICP [49], GLIM [50], LeGO-LOAM [51], COIN-LIO [52], LIO-SAM [53], Fast-LIO [54], CAE-LO [55], SSL-LO [56], DMLO [57], LO-Net [58], NDT-LOAM [39], E-LOAM [59].

sweep, points are reprojected by linear interpolation, assuming constant velocity — this is known as *motion/distortion compensation*. Mapping takes place in parallel, at a slower rate, and the current scan is registered to the map created so far. At that time, this implementation achieved the best results³ on the KITTI Odometry Benchmark [62]. V-LOAM [63] improved this solution by introducing a visual odometry component based on a fish-eye camera.

A few years later, LeGO-LOAM [51] extended the feature extraction process by performing segmentation on a range image generated from the 3D point cloud. Clusters are filtered by size, in order to discard features belonging to potentially noisy points. Another modification is that the parameters of the transformation between consecutive scans are optimized separately: $t_z, \theta_{roll}, \theta_{pitch}$ are computed using planar feature correspondences, then fixed during the optimization of t_x, t_y, θ_{yaw} , which only uses edge features. The method is compared to LOAM and achieves higher accuracy in outdoor scenarios, while being an order of magnitude faster.

F-LOAM [64] proposes a two-step motion compensation process. For odometry computation, the constant velocity model is used, but the points are re-corrected using the optimized pose before being registered to the map. Correspondences are weighted based on the “local smoothness” of the feature, and the non-linear optimization is solved using the Gauss-Newton method. The map is not updated at every scan, but only when the translational change reaches a predefined threshold. This is a common keyframe selection technique inherited from Visual Odometry.

A slightly different approach introduces the use of inertial sensors, leading to LiDAR-

³https://www.cvlibs.net/datasets/kitti/eval_odometry.php

2. Background and Literature Review

Inertial Odometry (LIO). This aims to compensate for the lack of reliable 3D features in specific environments, as accelerometers can provide satisfactory motion estimates for short displacements. FAST-LIO [54] applies such a technique for a drone equipped with a high-frequency solid-state LiDAR, by performing tightly-coupled fusion: instead of using the IMU output to correct the scan registration, it is applied on the features extracted from the point cloud. COIN-LIO [52] introduces a photometric error component, based on the beam intensity values returned by the LiDAR. A monochrome “intensity image” is constructed and filtered such that matching can occur between consecutive frames, and these new correspondences extend the residual vector that is minimized for odometry computation. The approach achieves state-of-the-art performance on a dataset of geometrically-degenerate scenes (ENWIDE ⁴).

In the class of deep-learning techniques, we highlight Deep Matching LiDAR Odometry (DMLO) [57], which translates the registration problem into a supervised machine learning task. Point clouds are projected into a 2D map using cylinder encoding, with range and intensity values as channels, and a Convolutional Neural Network (CNN) architecture is trained to predict correspondences from pairs of projections. Training samples are constructed from a subset of the target dataset, and the method proves robust across different LiDAR hardware, but does not surpass LOAM on the KITTI sequences. In comparison to [58], this solution does not leave the geometric problem to the inner workings of the CNN.

The approach that our work draws most inspiration from is KISS-ICP [45], a LiDAR-only odometry framework built around point-to-point ICP. This method proposes a few modifications that cooperate towards a hardware-agnostic solution, with a small number of adjustable parameters. The first contribution is a constant velocity motion estimation, which provides the optimization step with an initial guess. The same motion estimation is used for distortion compensation. Secondly, feature extraction is replaced by two stages of voxel-based down-sampling: the first down-sampled point cloud is used to extend the map, while the even lower-resolution set is registered against the existing map to compute the pose estimate. Perhaps the key component is an adaptive distance threshold for correspondence outlier removal. This threshold is updated based on the deviation between the predicted and optimized pose, acting as a form of uncertainty estimation. Additionally, the optimization problem employs a robust kernel, whose scale parameter is related to the adaptive threshold.

⁴<https://projects.asl.ethz.ch/datasets/enwide>

Chapter 3

Methodology

This chapter will address the core contributions of our project. We present the hardware involved in sensor fusion, describe the data acquisition procedure, then introduce our final solution, by detailing the research and implementation process, as well as design decisions that had to be made along the way.

3.1 Prerequisites

Let us first introduce the mathematical notions used throughout this work. The group of rotations in 3D space is $\text{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3 \times 3} : \det \mathbf{R} = 1, \mathbf{R}^T \mathbf{R} = \mathbf{I}_3\}$. For any vector $\mathbf{w} \in \mathbb{R}^3$ we can define a *skew-symmetric* matrix:

$$\hat{\mathbf{w}} = [\mathbf{w}]_{\times} = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \quad (3.1)$$

and $\mathfrak{so}(3) = \{\hat{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^3\}$ is the set of all such matrices. The exponential maps $\exp : \mathfrak{so}(3) \rightarrow \text{SO}(3)$ and $\text{Exp} : \mathbb{R}^3 \rightarrow \text{SO}(3)$ are defined as $\exp(\hat{\mathbf{w}}) = \text{Exp}(\mathbf{w}) = e^{\hat{\mathbf{w}}}$, and Rodrigues' formula [65] provides a closed formulation of the matrix exponential:

$$e^{\hat{\mathbf{w}}} = \mathbf{I}_3 + \frac{\hat{\mathbf{w}}}{|\mathbf{w}|} \sin |\mathbf{w}| + \frac{\hat{\mathbf{w}}^2}{|\mathbf{w}|^2} (1 - \cos |\mathbf{w}|). \quad (3.2)$$

3. Methodology

The inverse operations are $\log : \text{SO}(3) \rightarrow \mathfrak{so}(3)$ and $\text{Log} : \text{SO}(3) \rightarrow \mathbb{R}^3$, respectively. If $\hat{\mathbf{w}} = \log(\mathbf{R})$ and $\mathbf{R} \neq \mathbf{I}_3$, then:

$$|\mathbf{w}| = \cos^{-1} \left(\frac{\text{tr}(\mathbf{R}) - 1}{2} \right) \text{ and } \frac{\mathbf{w}}{|\mathbf{w}|} = \frac{1}{2 \sin |\mathbf{w}|} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix}, \quad (3.3)$$

and if $\mathbf{R} = \mathbf{I}_3$, $|\mathbf{w}| = 0$.

Furthermore, we consider the group of Euclidean transformations in 3D space (rotation and translation):

$$\text{SE}(3) = \left\{ \mathbf{M} \in \mathbb{R}^{4 \times 4} : \mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \mathbf{R} \in \text{SO}(3), \mathbf{t} \in \mathbb{R}^3 \right\}. \quad (3.4)$$

The corresponding tangent space is:

$$\mathfrak{se}(3) = \left\{ \hat{\xi} \in \mathbb{R}^{4 \times 4} : \hat{\xi} = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{w}} & \mathbf{v} \\ \mathbf{0}^T & 1 \end{bmatrix}, \mathbf{w} \in \mathbb{R}^3, \mathbf{v} \in \mathbb{R}^3 \right\}. \quad (3.5)$$

The exponential maps $\exp : \mathfrak{se}(3) \rightarrow \text{SE}(3)$ and $\text{Exp} : \mathbb{R}^6 \rightarrow \text{SE}(3)$ are defined as:

$$\exp(\hat{\xi}) = \text{Exp}(\xi) = \begin{bmatrix} \text{Exp}(\mathbf{w}) & \mathbf{V}(\mathbf{w})\mathbf{v} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad (3.6)$$

where:

$$\mathbf{V}(\mathbf{w}) = \mathbf{V}(\theta \mathbf{u}) = \mathbf{I}_3 + \frac{1 - \cos \theta}{\theta} \hat{\mathbf{u}} + \frac{\theta - \sin \theta}{\theta} \hat{\mathbf{u}}^2. \quad (3.7)$$

The logarithmic map $\text{Log} : \text{SE}(3) \rightarrow \mathbb{R}^6$ is:

$$\xi = \text{Log} \left(\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \right) = \begin{bmatrix} \mathbf{V}^{-1}(\mathbf{w})\mathbf{t} \\ \text{Log}(\mathbf{R}) \end{bmatrix}, \quad (3.8)$$

and we express $\log : \text{SE}(3) \rightarrow \mathfrak{se}(3)$ as $\hat{\xi} = \log(\mathbf{M}) = \widehat{\text{Log}(\mathbf{M})}$. This allows introducing an interpolation function, with a parameter $\alpha \in [0, 1]$:

$$\varphi(\mathbf{M}_1, \mathbf{M}_2, \alpha) := \mathbf{M}_1 \exp \left(\alpha \cdot \log(\mathbf{M}_1^{-1} \mathbf{M}_2) \right). \quad (3.9)$$

3.2 Hardware

In this context, hardware refers to the set of sensors used for data collection, determined by the existing industrial setup. The data capturing system is controlled by an

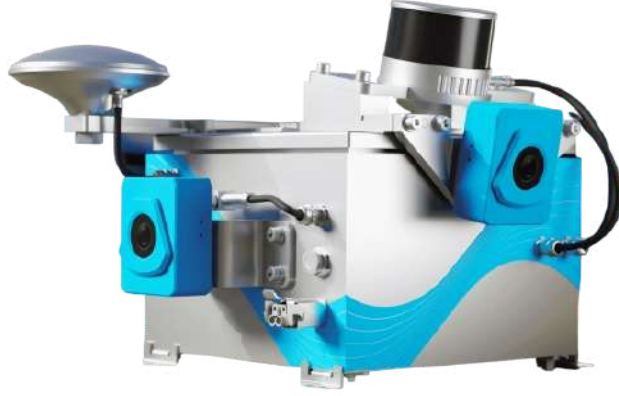


Figure 3.1: The SDX-Compact manufactured by Sodex Innovations GmbH. The set of sensors consists of a 3D LiDAR scanner, three RGB cameras and a high-accuracy positioning system. Image source: GeoLanes

Nvidia Jetson board, which operates as a middle-man for time synchronisation between the sensors, and coordinates the various data streams. Even though the solution was designed such that it does not inherently rely on any particular device, the relationship between hardware capabilities, data quality and final output makes it crucial to understand the sensors involved in the process. Beyond the components described below, the setup includes three industrial grade ArkCam Basic+ wide angle cameras which provide a 1920x1080 RGB stream over Ethernet. These are not utilised within this project, but represent a noticeable motivation for future work directions.

3.2.1 LiDAR Sensor

The SDX-Compact (Fig. 3.1) is equipped with a Pandar XT32 [66] LiDAR sensor, manufactured by Hesai Technology. This is a mechanical rotating LiDAR with a full 360° horizontal field of view and 32 beams distributed vertically, at 1° resolution. With our settings, the sensor produces 10 complete scans per second, resulting in a horizontal resolution of 0.18°. The maximum operational range is 120m, decreasing to 50m for low-reflectivity targets. The official specifications state a typical accuracy of $\pm 1\text{cm}$, with precision $\pm 0.5\text{cm}$, in a static environment. For each beam, the strongest return is processed, leading to 640,000 points being generated per second. The high output bandwidth is handled by an Ethernet connection, over which points are sent as UDP packets. The sensor also supports PTP synchronisation, essential for high-quality sensor fusion.

The same device has been used for LiDAR odometry applications in the past [67], and

has similar specifications to other popular scanners, such as Velodyne VLP-32C, Ouster OS1-32 or Robosense Helios 32.

3.2.2 GNSS/INS receiver

Another component of the sensor stack is the Septentrio AsteRx SBi3 Pro+ GNSS/INS receiver [68], which provides global positioning and orientation data at a rate of 100Hz. Internally, this relies on two distinct mechanisms.

The localization information comes from a dual antenna GNSS module compatible with several GNSS constellations (e.g., GPS⁵, GLONASS, Galileo), to ensure optimal worldwide coverage. In standalone mode, the advertised typical accuracy is 1m, but the receiver also acts as an NTRIP (a protocol for differential GPS) client, gathering correction information, in order to achieve centimeter-level accuracy.

An Inertial Measurement Unit (IMU) module records acceleration data and provides the remaining orientation angles (roll, pitch, yaw) to compute the complete pose, in 6 Degrees of Freedom (DoF). This is integrated with the absolute GNSS measurements using the patented FUSE+ technology [69], resulting in an orientation error below 5-10°.

Like any system reliant on satellite communication, this will suffer significantly in situations where the signal propagation is disturbed (heavy clouds, “urban canyons”, thick vegetation, spoofing), even leading to loss of *GNSS fix*.

To conclude this section, we recognize and underline the importance of accurate extrinsic calibration between the LiDAR and the local INS coordinate frame, which has to be performed prior to any reliable data collection procedure. Given the radically different modalities of these two sensors, this is not a trivial task [70] and lies outside the scope of the current work.

3.3 Data acquisition and pre-processing

Data is collected by mounting the sensor rig on the roof of a vehicle, and driving around a target area at a relatively low speed (e.g., up to 40km/h), see Fig. 3.2. In comparison with other public datasets [71, 72], capturing data in non-urban environments introduces some challenges — scenes dominated by vegetation, with few identifiable fea-

⁵Throughout this paper, GNSS and GPS are used interchangeably to refer to the global localization sensor, regardless of the underlying satellite system used during data collection.

3. Methodology

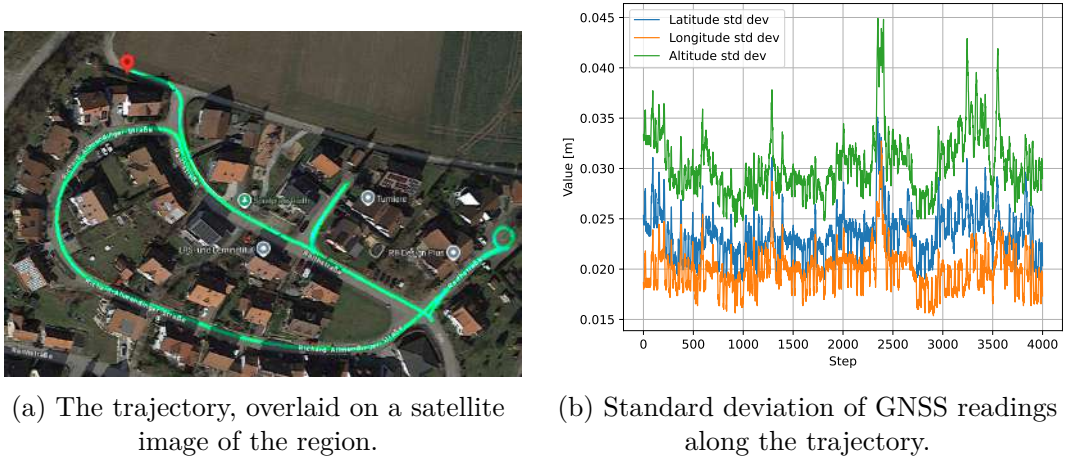


Figure 3.2: An example dataset collected in rural Germany (Lienziegen).

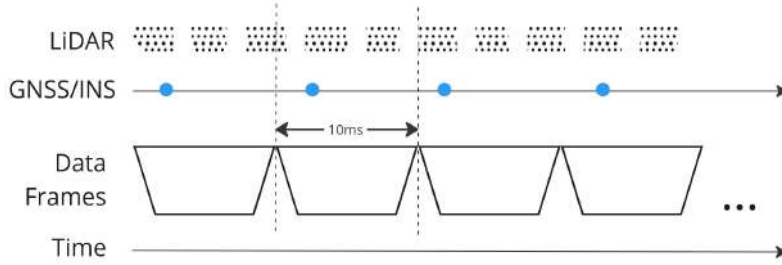


Figure 3.3: The data synchronisation process.

Time is discretized into fixed-size intervals, resulting in data frames of custom resolution.

These are populated with information from the two sensors: LiDAR points arrive in groups, as UDP packets, while GNSS readings have a frequency of approx. 100Hz. We employ a data frame size of 10ms to ensure that each interval has an associated GNSS measurement, and place all corresponding packets in the same data frame.

tures, uneven terrain — while minimizing others — negligible amount of dynamic objects, no tall buildings that could affect GNSS signal propagation.

The raw sensor output is uploaded to a cloud storage facility, and is later processed into *data frames* that fuse the available information (Fig. 3.3). This can occur because the sensors are PTP-synchronized on initialization, and all recorded data is timestamped.

The next pre-processing step consists of dividing the sequence of data frames such that we operate on individual scans (also known as sweeps) produced by the LiDAR. A scan corresponds to a complete 360° rotation, which takes 100ms, so we join the points in 10 consecutive data frames to obtain a single scan.

Every INS reading undergoes a map projection, to obtain x, y, z coordinates in the East-North-Up frame. We consider the frame of the GNSS receiver as the *ego* coordinate system. The roll ϕ , pitch θ and yaw angles ψ determine the absolute orientation, so we

3. Methodology

compute a global pose ${}^W\mathbf{T}_{ego} \in \text{SE}(3)$ as:

$${}^W\mathbf{T}_{ego} = \text{Translation}(x, y, z) \cdot \text{Rot}_z(\psi) \cdot \text{Rot}_x(\theta) \cdot \text{Rot}_y(\phi), \quad (3.10)$$

where $\text{Rot}_k(\alpha)$ is the transformation matrix corresponding to a rotation of α around axis k . Because the sensor provides error estimates in the form of global one-sigma values $\{\sigma_x, \sigma_y, \sigma_z, \sigma_\phi, \sigma_\theta, \sigma_\psi\}$, we construct the covariance matrix:

$$\Sigma = \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_z^2, \sigma_\phi^2, \sigma_\theta^2, \sigma_\psi^2) \quad (3.11)$$

and transform it using the adjoint map of the rotation component:

$$\Sigma_{ego} = \text{Adj}(\mathbf{R}, \mathbf{0}) \cdot \Sigma \cdot \text{Adj}(\mathbf{R}, \mathbf{0})^T, \quad (3.12)$$

where:

$$\text{Adj}(\mathbf{R}, \mathbf{t}) = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3 \times 3} \\ [\mathbf{t}]_{\times} \mathbf{R} & \mathbf{R} \end{bmatrix}. \quad (3.13)$$

A LiDAR range reading r , captured at azimuth α with an elevation angle of ϕ , can be converted to a 3D location in the LiDAR frame:

$${}^{\text{LiDAR}}\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} r \cos \phi \sin \alpha \\ r \cos \phi \cos \alpha \\ r \sin \phi \end{bmatrix}. \quad (3.14)$$

If ${}^{ego}\mathbf{T}_{\text{LiDAR}}$ is the pose of the LiDAR in the ego frame (from extrinsic calibration), we can compute the location of a point in the ego frame:

$$\begin{bmatrix} {}^{ego}\mathbf{p}_{\text{LiDAR}} \\ 1 \end{bmatrix} = {}^{ego}\mathbf{T}_{\text{LiDAR}} \begin{bmatrix} {}^{\text{LiDAR}}\mathbf{p} \\ 1 \end{bmatrix}. \quad (3.15)$$

3.3.1 Motion compensation

At this stage, it is worth discussing the distortion effect that occurs when a rotating LiDAR sensor is moved at a relatively high velocity. Because it operates in a relative coordinate frame and different beams of the sweep are fired at different times, the beams corresponding to azimuth 0° will fire approximately 100ms earlier than the beams corresponding to azimuth 359° . Placing the resulting points in the same coordinate frame would lead to undesired artifacts, such as duplicate or warped structures. (Fig. 3.4a)

This open research problem [73] is commonly addressed by estimating the sensor pose change during a sweep [74, 45], and IMU integration proves satisfactory [75], given

3. Methodology

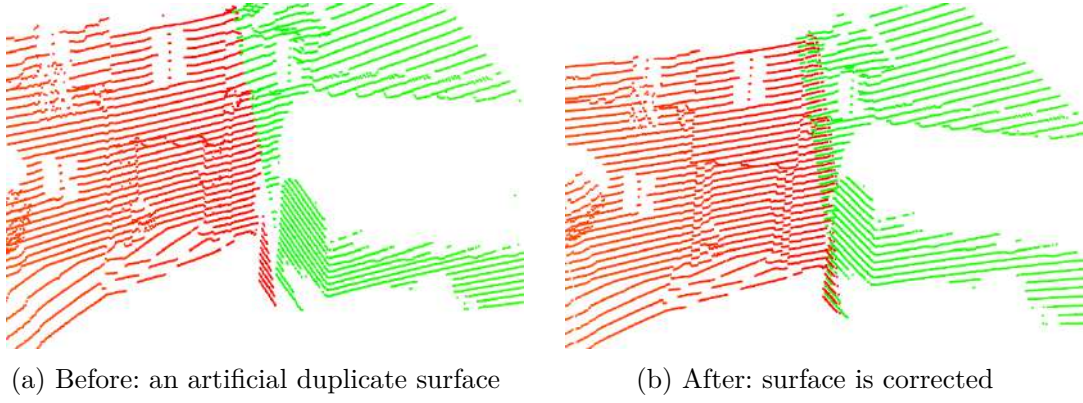


Figure 3.4: Motion artifacts and the result of motion compensation.

Green: points from the beginning of the sweep. Red: points from the end of the sweep. Without motion compensation, the vertical surface yields two conflicting clusters, so the scan cannot be used for accurate mapping in the global frame.

the short time interval involved. In our case, the ego poses $\{{}^W\mathbf{T}_{i,0}, {}^W\mathbf{T}_{i,1}, \dots, {}^W\mathbf{T}_{i,m}\}$ computed from INS measurements during sweep i are used to bring all points into the frame defined by ${}^W\mathbf{T}_{i,0}$. A point ${}^{i,k}\mathbf{p}$ belongs to data frame k , so it will be replaced by:

$$\begin{bmatrix} {}^{i,0}\mathbf{p} \\ 1 \end{bmatrix} = {}^{i,0}\mathbf{T}_{i,k} \begin{bmatrix} {}^{i,k}\mathbf{p} \\ 1 \end{bmatrix} = ({}^W\mathbf{T}_{i,0})^{-1} {}^W\mathbf{T}_{i,k} \begin{bmatrix} {}^{i,k}\mathbf{p} \\ 1 \end{bmatrix}. \quad (3.16)$$

We also experimented with approaches that do not rely on the absolute pose measurements for subsections of the sweep, such as interpolation using point timestamps or point indices (based on the order in which the points are returned), but these did not yield better results. A potential drawback of our method is that it disregards the localization noise, which could prove counterproductive if the GNSS receiver has low accuracy.

Through this step we are effectively removing the need for sub-scan information and creating a simpler data structure in which each scan is associated a single ${}^W\mathbf{T}_{ego}$ pose (from the first data frame), and the points in a sweep can be treated as a unified set.

Without loss of generality, we transform the sequence of global poses $\{{}^W\mathbf{T}_0, {}^W\mathbf{T}_1, \dots, {}^W\mathbf{T}_n\}$ into the frame of the first pose, by multiplying each pose with $({}^W\mathbf{T}_0)^{-1}$, to obtain $\{\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_n\}$. Naturally, $\mathbf{T}_0$ will always be $\mathbf{I}_4$, which simplifies the initialization of the odometry estimation. The covariance of each pose is adjusted with the adjoint of $({}^W\mathbf{T}_0)^{-1}$.

3.4 Solution architecture

At its core, the odometry and mapping solution that we propose requires minimal input: a sequence of point clouds captured by a moving LiDAR scanner, and a timestamp for each scan. If available, absolute localization/INS measurements can be used as additional input.

Let $\mathfrak{P} = \{P_0, P_1, \dots, P_n\}$ represent the set of point clouds that we operate on, with point coordinates $P_k = \{\mathbf{p}_{k,i} : \mathbf{p}_{k,i} \in \mathbb{R}^3\}$ expressed in the local frame, $\mathfrak{s} = \{s_0, s_1, \dots, s_n\}$ the corresponding timestamps, and $\hat{\mathfrak{T}} = \{\hat{\mathbf{T}}_0, \hat{\mathbf{T}}_1, \dots, \hat{\mathbf{T}}_n\}$ the ground truth poses at which each scan was captured. In an ideal setting, a point cloud registration algorithm would take P_i and P_{i+1} as input and return the transformation $\Delta\hat{\mathbf{T}}_i = \hat{\mathbf{T}}_i^{-1}\hat{\mathbf{T}}_{i+1}$, i.e., the ground truth displacement between consecutive poses. This would correspond to a perfect odometry model, and the complete trajectory could be reconstructed by accumulating the computed displacement. Unfortunately, with the exception of some simulation environments, such an approach is not actually feasible. As we have already seen, LiDAR output is not perfect, especially in dynamic scenes, and some scenarios are simply unsuitable for odometry estimation based on 3D features [76].

The output of our solution can be formulated as $\mathfrak{T} = \{\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_n\}$. Pose $\mathbf{T}_k \in \text{SE}(3)$ represents the estimated location and orientation of the *ego* frame from which the points P_k were observed. The complete *map* estimate, represented as a larger point cloud in global coordinates, is the set of all points, each transformed according to their respective pose:

$$M(\mathfrak{T}, \mathfrak{P}) = \left\{ \mathbf{p}_{k,i}^* \in \mathbb{R}^3 : \begin{bmatrix} \mathbf{p}_{k,i}^* \\ 1 \end{bmatrix} = \mathbf{T}_k \begin{bmatrix} \mathbf{p}_{k,i} \\ 1 \end{bmatrix}, \mathbf{p}_{k,i} \in P_k, k \in \overline{0 \dots n} \right\}. \quad (3.17)$$

This uncovers two possible evaluation directions, which also constitute the high-level aims of our method. On one hand, the output trajectory $\mathfrak{T}$ should match the actual path of the system as accurately as possible. On the other, the final map $M(\mathfrak{T}, \mathfrak{P})$ should not hide or damage small details in the scene, to obtain a high-quality 3D reconstruction. Although not immediately obvious, these can result in conflicting requirements, but they anticipate an underlying optimization problem.

3. Methodology

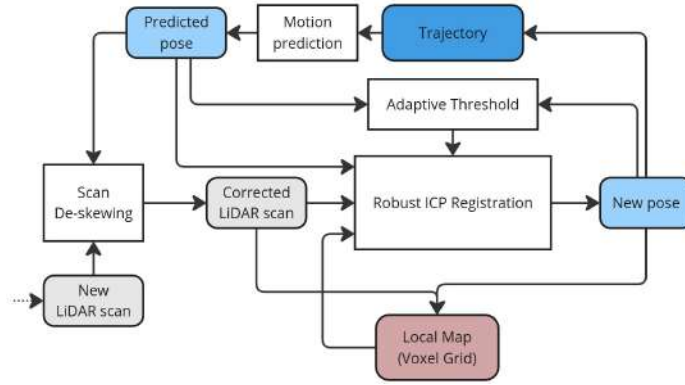


Figure 3.5: The KISS ICP pipeline.

An incoming scan is motion-compensated using the displacement between the previous estimated poses, then it is downsampled and registered against the local map. The pose estimated by the ICP algorithm is added to the trajectory, and the adaptive threshold is updated by evaluating the accuracy of the motion estimation. This determines the ICP outlier removal threshold, improving the system’s ability to react to unexpected motions.

3.4.1 Overview

The final architecture is the result of a series of experiments. The process began with a LiDAR-only framework, imitating the KISS-ICP [45] implementation, shown in Fig. 3.5. In this approach, scans are processed one-by-one, attempting to compute the optimal pose at each step, and the map is built incrementally. This is computationally efficient and becomes essential when a robot operates in an unknown environment, as it provides a localization estimate at any given time. In the absence of absolute positioning information, however, this diverges from the ground-truth trajectory (Fig. 3.6), creating a distorted map.

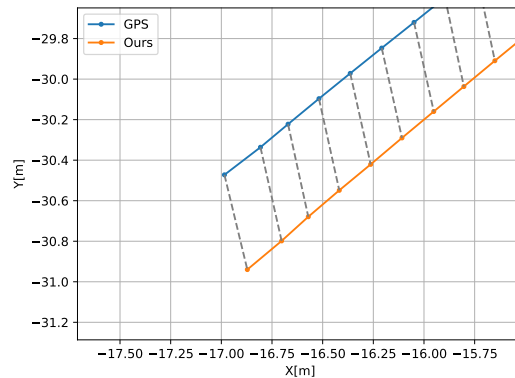


Figure 3.6: When the GPS information is not used, the trajectory computed based on LiDAR data diverges. Here, after 300 scans (approx. 30 seconds), the last location is approx. 0.5m away from the corresponding GPS position.

3. Methodology

Early efforts focused on introducing the GPS readings as a constraint at each step, alongside motion prediction and registration results. This led to experimenting with each component, as well as methods for combining the respective poses, while regularly monitoring the quality of the output trajectories and maps. As the solution was evolving, it was tested on more challenging scenarios, by adding artificial noise to the GPS readings, or skipping some of them altogether.

Later on, we also considered addressing the pose estimation problem holistically, instead of in an incremental fashion, and experimented with architectures based on factor graph optimization [77], where nodes are linked using registration and GPS constraints. The same building blocks are needed, but the relationship between them is guided by the graph paradigm. Eventually, this was adapted to obtain a hybrid solution (Fig. 3.13), which combines the advantages of the two frameworks in a very intuitive manner.

3.4.2 Reconciling motion prediction, GNSS and registration

Before discussing the components in detail, we would like to provide a general picture of the rationale behind combining the pose estimates that we operate on. We assume that for every scan P_k we can compute three poses in the world frame, and aim to generate $\mathbf{T}_k$ as a function of these:

1. A registration-predicted pose $\mathbf{T}_k^{\text{ICP}}$, from a point cloud registration method that optimizes map alignment between scans in the global frame.
2. An absolute localization pose $\mathbf{T}_k^{\text{GPS}}$, generated by the INS sensor.
3. A motion-predicted pose $\mathbf{T}'_k$, output by the algorithm described in Section 3.4.3.

Each of these serves a different purpose: using only $\mathbf{T}_k^{\text{ICP}}$ would lead to a high-quality 3D map, while diverging from the actual locations; $\mathbf{T}_k^{\text{GPS}}$ estimates the absolute position and orientation, but its accuracy varies depending on the environment; the $\mathbf{T}'_k$ pose can help filter out noise in the GNSS data and smoothen the trajectory, but comes from a motion prediction mechanism that cannot operate in the absence of other sensors.

In an initial trial, we observed what happens when the GPS pose is used as the initial state of the registration algorithm, with the intuition that this could prevent trajectory divergence without significantly affecting map quality. In practice, this leads to an unwanted behavior. At first, the registration is not affected by the input pose, converging to the pure $\mathbf{T}_k^{\text{ICP}}$. As the map grows, the divergence is not constrained, so it reaches a stage where the $\mathbf{T}_k^{\text{GPS}}$ pose is a very poor initial guess for registration, which gets stuck in a

3. Methodology

local minimum. Depending on the displacement between scans and the nature of the 3D scenes, this can even happen after just a few scans.

A subsequent idea was to treat $\mathbf{T}_k^{\text{ICP}}$ as local refinement for $\mathbf{T}_k^{\text{GPS}}$, by applying an interpolation, as defined in Eq. 3.9. If $\mathbf{T}_k = \varphi(\mathbf{T}_k^{\text{GPS}}, \mathbf{T}_k^{\text{ICP}}, \alpha)$, the interpolation parameter α controls how much the registration pose is allowed to alter the GPS prior — when $\alpha = 1$, we obtain the pure registration-based trajectory. This approach can prevent trajectory divergence while improving map quality, for small values of α (e.g., 0.1). However, since α is fixed, it does not address the problem of varying GPS noise or faulty registration results.

This can also be formulated as a generic optimization problem. First, we define a distance measure between an arbitrary pose $\mathbf{T}$ and a target pose $\mathbf{M}$:

$$\mathcal{D}(\mathbf{T}, \mathbf{M}, \Sigma_{\mathbf{M}}) = \left(\text{Log}(\mathbf{T}^{-1}\mathbf{M}) \right)^{\text{T}} \cdot \Sigma_{\mathbf{M}}^{-1} \cdot \text{Log}(\mathbf{T}^{-1}\mathbf{M}), \quad (3.18)$$

where $\Sigma_{\mathbf{M}}$ is the covariance of $\mathbf{M}$, and $\text{Log}(\mathbf{T}^{-1}\mathbf{M}) \in \mathbb{R}^6$ is the twist vector residual. What we aim to optimize is:

$$\mathcal{J}(\mathbf{T}) = \beta_{\text{Reg}} \mathcal{J}_{\text{Reg}}(\mathbf{T}) + \beta_{\text{GPS}} \mathcal{D}(\mathbf{T}, \mathbf{T}_k^{\text{GPS}}, \Sigma_{\text{GPS}}) + \beta_{\text{Motion}} \mathcal{D}(\mathbf{T}, \mathbf{T}'_k, \Sigma'), \quad (3.19)$$

where $\mathcal{J}_{\text{Reg}}$ defines the cost associated with the registration, and the β weights balance the influence of the components. If we assume that $\mathbf{T}'_k$ already includes the information from $\mathbf{T}_k^{\text{GPS}}$, the middle term can be discarded.

In one of our experiments, we expressed $\mathcal{J}_{\text{Reg}}$ as the L2 norm of the vector of point cloud registration residuals. In classical ICP, these are pairwise distances between points that have been matched using a nearest-neighbor search. However, optimizing this jointly with the other pose-based residuals (from $\mathcal{D}$) leads to numerical complications, because of the different scales of the errors involved. We note that this could represent a future research direction.

Another approach is to see $\mathbf{T}_k^{\text{ICP}}$ as the optimal registration pose and have

$$\mathcal{J}_{\text{Reg}}(\mathbf{T}) = \mathcal{D}(\mathbf{T}, \mathbf{T}_k^{\text{ICP}}, \Sigma_{\text{ICP}}), \quad (3.20)$$

where Σ_{ICP} is a covariance matrix that estimates the registration uncertainty — this is explained in more detail in Section 3.4.4. The cost function becomes a weighted sum of

3. Methodology

$\mathcal{D}$ functions, so the optimization problem at time step k is:

$$\mathbf{T}_k = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{\mathbf{M}} \beta_{\mathbf{M}} \mathcal{D}(\mathbf{T}, \mathbf{M}, \Sigma_{\mathbf{M}}), \quad (3.21)$$

where $\mathbf{M} \in \{\mathbf{T}_k^{\text{ICP}}, \mathbf{T}_k^{\text{GPS}}, \mathbf{T}'_k\}$. As this is intractable on the non-linear manifold SE(3), an approximation of the solution can be computed using the Weiszfeld algorithm (Alg. 1). We disregard the scalar weights $\beta_{\mathbf{M}}$, as they can be accounted for, numerically, in the covariances $\Sigma_{\mathbf{M}}$.

Algorithm 1 Weiszfeld algorithm for minimizing a custom pose distance metric.

Input:

- $\mathcal{M}$: list of poses $\mathbf{M}$ with associated covariance $\Sigma_{\mathbf{M}}$
- $\mathbf{T}_0$: initial pose estimate
- $i_{\max}$: upper iteration threshold for the algorithm

Output:

- $\mathbf{T}$: approximation of the pose that minimizes $\sum_{\mathbf{M}} \mathcal{D}(\mathbf{T}, \mathbf{M}, \Sigma_{\mathbf{M}})$

Funct Weiszfeld($\mathcal{M}, \mathbf{T}_0, i_{\max}$)

```

1:  $\mathbf{T} \leftarrow \mathbf{T}_0$ 
2: for  $i = 1, \dots, i_{\max}$  do
3:    $w_{\xi} \leftarrow \mathbf{0}_{6 \times 1}$ 
4:    $w \leftarrow 0$ 
5:   for each  $\mathbf{M} \in \mathcal{M}$  do
6:      $d_{\mathbf{M}} \leftarrow \max(\mathcal{D}(\mathbf{T}, \mathbf{M}, \Sigma_{\mathbf{M}}), 10^{-7})$   $\triangleright$  Apply threshold to avoid singularities
7:      $w_{\xi} \leftarrow w_{\xi} + \operatorname{Log}(\mathbf{T}^{-1}\mathbf{M}) d_{\mathbf{M}}^{-1}$ 
8:      $w \leftarrow w + d_{\mathbf{M}}^{-1}$ 
9:   end for
10:  if  $w < 10^{-7}$  then
11:    break  $\triangleright$  Early stop, degenerate weight sum
12:  end if
13:   $\xi = w_{\xi}/w$ 
14:  if  $\|\xi\|_2 < 10^{-7}$  then
15:    break  $\triangleright$  Early stop, no significant update occurred
16:  end if
17:   $\mathbf{T} \leftarrow \operatorname{TExp}(\xi)$ 
18: end for
19: return  $\mathbf{T}$ 
```

Alternatively, we observe that Eq. 3.21 involves a covariance-weighted sum of residuals, so another approximation of the solution is obtained from the covariance-weighted average in the twist vector space:

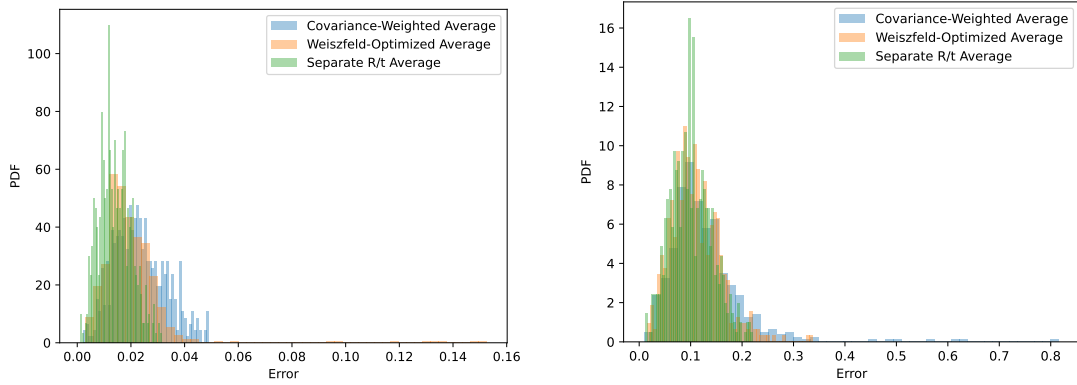
$$\xi = \left(\sum_{\mathbf{M}} \Sigma_{\mathbf{M}}^{-1} \right)^{-1} \sum_{\mathbf{M}} \Sigma_{\mathbf{M}}^{-1} \operatorname{Log}(\mathbf{T}_0^{-1}\mathbf{M}), \quad (3.22)$$

3. Methodology

where $\mathbf{T} = \mathbf{T}_0 \text{Exp}(\xi)$ and $\mathbf{T}_0$ is a linearization pose.

In Structure from Motion (SfM) tasks [78], pose averaging is split into rotation and translation averaging. For a set of matrices $\mathcal{R} = \{\mathbf{R}_i \in \text{SO}(3)\}$ with weights $\mathcal{W} = \{w_i \in \mathbb{R}\}$, we can first perform Singular Value Decomposition (SVD), $\mathbf{UDV}^T = \sum_i w_i \mathbf{R}_i$. The *chordal mean* [79] is then obtained as $\bar{\mathbf{R}} = \mathbf{UV}^T$. The scalar weights must be approximated from the covariance matrix, e.g., $w_i = \text{tr}(\Sigma_{\mathbf{R}_i})$. For translations, we exploit the linearity properties and compute $\bar{\mathbf{t}} = \left(\sum_{\mathbf{t}} \Sigma_{\mathbf{t}}^{-1}\right)^{-1} \sum_{\mathbf{t}} \Sigma_{\mathbf{t}}^{-1} \mathbf{t}$.

These three approaches have been compared by generating large sets of poses with known covariance around an arbitrary reference pose (treated as the ground truth mean), and assessing how the predicted mean differs from it. The results of this experiment are presented in Fig. 3.7. We observed no significant effect of either of these methods on the final trajectory, so the covariance-weighted approximation was used, thanks to its relatively simple formulation.



(a) Distribution of average translation error. (b) Distribution of average rotation error.

Figure 3.7: Comparing the behaviour of three different pose averaging techniques. We generate a random pose (expected mean), then sample 100 poses around it, by sampling a normal distribution of known σ for each parameter. The sample poses are averaged using the target algorithm, and we compute a translation and a rotation error:

$$e(\mathbf{t}_1, \mathbf{t}_2) = \|\mathbf{t}_1 - \mathbf{t}_2\|_2 \text{ and } e(\mathbf{R}_1, \mathbf{R}_2) = \left\| \text{Log}(\mathbf{R}_1^{-1} \mathbf{R}_2) \right\|_2, \text{ respectively.}$$

3.4.3 Motion prediction

The first component that we analyse addresses the problem of predicting the next pose in the trajectory. Given the trajectory computed so far $\{\mathbf{T}_0, \mathbf{T}_1, \dots, \mathbf{T}_t\}$, we are looking for $\mathbf{T}'_{t+1}$ that estimates the location and orientation of scan P_{t+1} in the world frame.

This influences two other elements of the framework: motion compensation and point cloud registration. In the absence of IMU data, the displacement between consecutive

3. Methodology

scans can be interpolated to correct individual points. Point cloud registration uses this predicted pose as an initial guess, for faster convergence, as we will see in a later section.

If the scans are captured at small, equal intervals, as is usually the case for LiDAR sensors, the movement can be approximated by a constant velocity model. The displacement between the previous two scans is propagated using $\Delta \mathbf{T}'_t \approx \Delta \mathbf{T}_{t-1} = \mathbf{T}_{t-1}^{-1} \mathbf{T}_t$, which leads to $\mathbf{T}'_{t+1} = \mathbf{T}_t \Delta \mathbf{T}_{t-1}$. In our case, the time steps are not necessarily equal, and we would like to correct this pose estimate using the GPS measurements, when available, so we experimented with a Kalman Filter for computing the location component of $\mathbf{T}'_{t+1}$. The state $\mathbf{x}_k \sim \mathcal{N}(\bar{\mathbf{x}}_k, \mathbf{P}_k)$ is formulated as:

$$\bar{\mathbf{x}}_k = [x_k, y_k, z_k, v_{x_k}, v_{y_k}, v_{z_k}]^T \quad (3.23)$$

$$\mathbf{P}_k = \text{diag}(\sigma_{x_k}^2, \sigma_{y_k}^2, \sigma_{z_k}^2, \sigma_{v_{x_k}}^2, \sigma_{v_{y_k}}^2, \sigma_{v_{z_k}}^2). \quad (3.24)$$

Over a time step Δt , this changes according to a linear model:

$$\hat{\bar{\mathbf{x}}}_{k+1} = \mathbf{F}(\Delta t) \cdot \bar{\mathbf{x}}_k \quad (3.25)$$

$$\hat{\mathbf{P}}_{k+1} = \mathbf{F}(\Delta t)^T \cdot \mathbf{P}_k \cdot \mathbf{F}(\Delta t) + \mathbf{Q}(\Delta t), \quad (3.26)$$

where $\mathbf{F}(\Delta t) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{I}_3 \Delta t \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$ and $\mathbf{Q}(\Delta t)$ is the estimated process noise, which scales with the time interval. In the update step, we compute the new mean and covariance using a measurement $\mathbf{z}_k \sim \mathcal{N}(\mathbf{z}_k, \mathbf{R})$, by applying a Kalman gain:

$$\mathbf{K} = \hat{\mathbf{P}}_{k+1} \mathbf{H} (\mathbf{H} \hat{\mathbf{P}}_{k+1} \mathbf{H}^T + \mathbf{R})^{-1} \quad (3.27)$$

$$\bar{\mathbf{x}}_{k+1} = \hat{\bar{\mathbf{x}}}_{k+1} + \mathbf{K} (\mathbf{z}_k - \mathbf{H} \hat{\bar{\mathbf{x}}}_k) \quad (3.28)$$

$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K} \mathbf{H}) \hat{\mathbf{P}}_{k+1}. \quad (3.29)$$

The $\mathbf{z}_k$ vector consists of a location $[x_{\text{GPS}}, y_{\text{GPS}}, z_{\text{GPS}}]^T$ and velocity values $\frac{1}{\Delta t} [x_{\text{GPS}} - x_k, y_{\text{GPS}} - y_k, z_{\text{GPS}} - z_k]^T$, assuming linear displacement. The Jacobian $\mathbf{H}$ with respect to the state vector is $\mathbf{I}_6$, and the covariance of the measurement noise $\mathbf{R}$ can be estimated using the covariance of the GNSS reading. If no GNSS reading is associated with the current scan, the pose estimate is retrieved from the predicted position $\hat{\bar{\mathbf{x}}}_{k+1}$, and the update step is performed based on the pose computed by the registration procedure.

This type of motion prediction method suffers from several disadvantages which outweigh its benefits for our use case. Firstly, it does not address the rotation component

3. Methodology

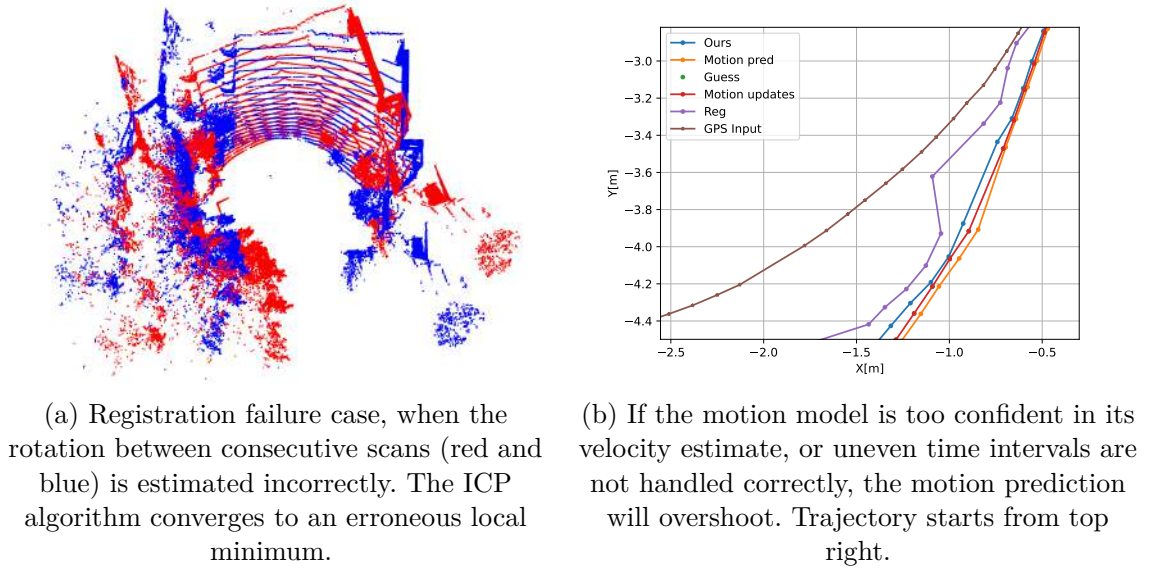


Figure 3.8: Challenges of using a Kalman Filter motion model.

of the poses. Some possible workarounds include copying the orientation from the previous pose and letting the registration component correct that — this has been observed to fail when a larger time gap occurs, especially for curved trajectories (Fig. 3.8a) —, using the rotation returned by the INS, or having a separate method for computing the new rotation. Secondly, it introduces the need for reliable noise estimation (Fig. 3.8b), such that the model is reactive to unexpected movement, while filtering out measurement noise. Process noise covariance $\mathbf{Q}(\Delta t)$ and the measurement input covariance $\mathbf{R}$ require approximations which are difficult to balance for a general solution. Therefore, using the estimated uncertainty of the predicted pose in further computations would only propagate potential sources of error.

The approach that we propose relies on the interpolation properties of the $\text{SE}(3)$ manifold (Eq. 3.9). In our case, using $\mathbf{T}_{t-1}$ and $\mathbf{T}_t$ to compute $\mathbf{T}'_{t+1}$ is actually a matter of extrapolation, because we expect α to be approximately 2 at most steps, which would lead to unstable behaviour in the Lie group transformation. To avoid this, we first compute $\mathbf{T}''_{t+1} = \mathbf{T}_t \mathbf{T}_{t-1}^{-1} \mathbf{T}_t$, and then obtain $\mathbf{T}'_{t+1} = \varphi(\mathbf{T}_t, \mathbf{T}''_{t+1}, \alpha_t - 1)$, where $\alpha_t = \frac{s_{t+1} - s_{t-1}}{s_t - s_{t-1}}$. For the scenarios that we have tested on, this provides a very good approximation of the correct displacement, which can be further refined using the information in the point cloud.

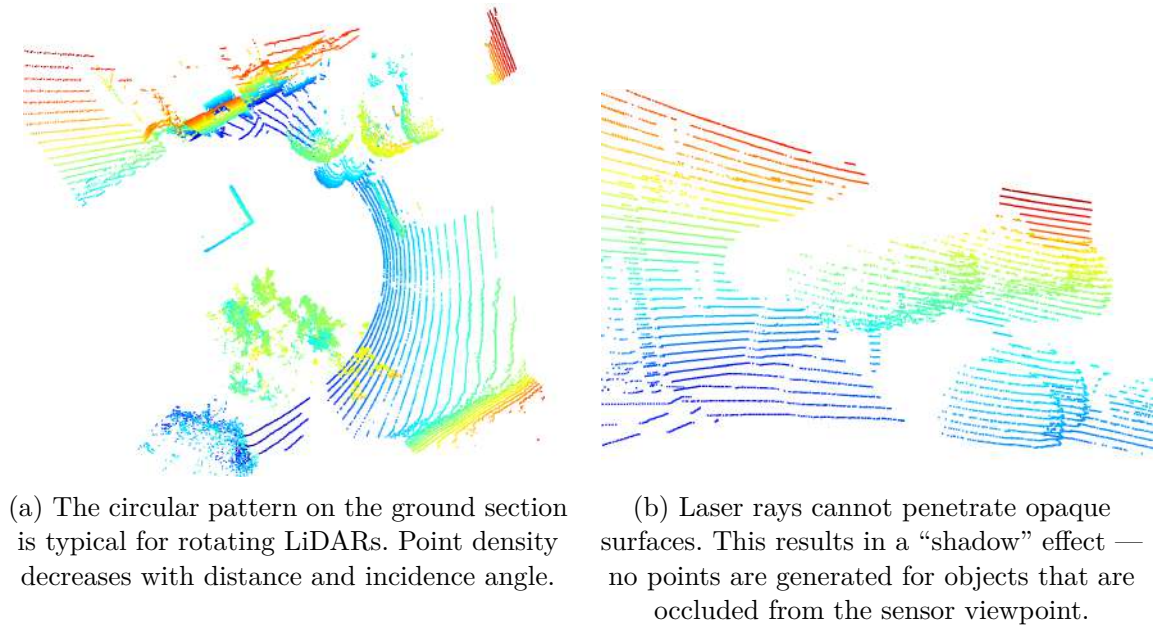


Figure 3.9: Particularities of LiDAR output.

3.4.4 Point Cloud Registration

The LiDAR sensor perceives the environment in a digital, discrete way, and outputs a set of 3D coordinates in the local frame. In a static scene, or after accurate motion compensation (Section 3.3.1), this represents a high-quality 3D snapshot of the environment from the current viewpoint. The mechanical design of the sensor introduces some particularities in the data (Fig. 3.9), but the measurement accuracy is excellent. As described earlier, applying a noise-resilient registration algorithm between consecutive scans constitutes a sensible odometry estimate.

The ICP algorithm

A baseline for *fine* registration methods, Iterative Closest Point (Algorithm 2) is naturally applicable to our scenario, albeit with some modifications. As opposed to *coarse* registration, where the aim is to find a rough alignment between arbitrary point clouds, we can exploit the motion prediction and/or the GPS readings as prior information about their global pose.

At every iteration step, the algorithm generates a set of correspondence hypotheses — a heuristic approximation of true correspondences — by finding the closest point from Q to each point in P' . The assumption is that the alignment of the two point clouds is improved by minimizing the distance associated with each correspondence. Depending on

3. Methodology

Algorithm 2 Pseudocode of the point-to-point ICP algorithm.

Input:

- P, Q : two point clouds, in local coordinate frames
- $\mathbf{T}_{\text{init}}$: initial transformation guess between P and Q
- d_{max} : upper distance threshold for point neighbor search
- i_{max} : upper iteration threshold for the algorithm

Output:

- $\mathbf{T}$: registration pose between P and Q

Funct ICP($P, Q, \mathbf{T}_{\text{init}}, d_{\text{max}}, i_{\text{max}}$)

```

1:  $\mathbf{T} \leftarrow \mathbf{T}_{\text{init}}$ 
2:  $P' \leftarrow P$  transformed with  $\mathbf{T}$ 
3: for  $i = 1, \dots, i_{\text{max}}$  do
4:   Compute nearest-neighbor correspondences between  $P'$  and  $Q$ 
5:   Discard correspondences where  $\|p' - q\|_2 > d_{\text{max}}$ 
6:   Compute  $\mathbf{T}' \leftarrow \underset{\mathbf{T}}{\operatorname{argmin}} \sum_{p', q} \left\| \mathbf{T} \begin{bmatrix} p' \\ 1 \end{bmatrix} - \begin{bmatrix} q \\ 1 \end{bmatrix} \right\|_2$ 
7:    $\mathbf{T} \leftarrow \mathbf{T}'\mathbf{T}$  ▷ Update the transformation estimate
8:    $P' \leftarrow P'$  transformed with  $\mathbf{T}'$  ▷ Update the point cloud
9: end for
10: return  $\mathbf{T}$ 

```

how the distance metric is defined, the algorithm can focus on different characteristics of the point clouds. The simplest formulation uses a point-to-point distance metric. Given a transformation estimate $\mathbf{T} = (\mathbf{R}, \mathbf{t})$, we have:

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{R}\mathbf{p} + \mathbf{t} - \mathbf{q}\|_2. \quad (3.30)$$

This is a computationally-efficient approach, but is more sensitive to noise and outliers compared to the alternatives, and it does not take into account any geometrical features of the point clouds. To improve this, a point-to-plane metric [27] can be used, which considers the unit surface normal $\mathbf{n}_q$ computed at $\mathbf{q}$:

$$d_2(\mathbf{p}, \mathbf{q}) = \left(\mathbf{n}_q^T (\mathbf{R}\mathbf{p} + \mathbf{t} - \mathbf{q}) \right)^2. \quad (3.31)$$

Because $d_2(\mathbf{p}, \mathbf{q})$ is null when $\mathbf{p}$ lies on the plane defined by $\mathbf{q}$ and $\mathbf{n}_q$, this approach improves surface registration, but will struggle in unstructured environments where the normals cannot be reliably estimated. In general, this is achieved by considering the neighborhood of each point in Q , fitting a plane to it, and computing its normal.

In a more general formulation, the local surface properties are captured by the covariance matrices $\mathbf{C}_p, \mathbf{C}_q$ computed at $\mathbf{p}$ and $\mathbf{q}$ respectively. The resulting distance metric [28] is:

3. Methodology

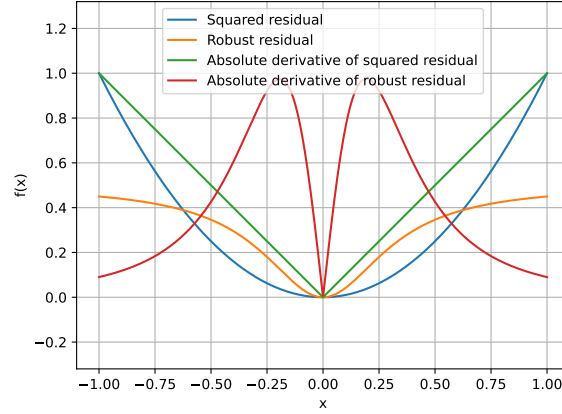


Figure 3.10: Applying the Geman-McClure function ($\kappa = 0.11$) to the correspondence residuals modifies their relative contribution during least-squares optimization, potentially improving convergence.

$$d_3(\mathbf{p}, \mathbf{q}) = (\mathbf{R}\mathbf{p} + \mathbf{t} - \mathbf{q})^T (\mathbf{R}\mathbf{C}_p\mathbf{R}^T + \mathbf{C}_q)^{-1} (\mathbf{R}\mathbf{p} + \mathbf{t} - \mathbf{q}). \quad (3.32)$$

All of the above can constitute the residual terms when computing the optimal transformation using a Least-Squares solver. To improve noise robustness, several additional techniques are employed. We prevent the creation of correspondences beyond a given distance threshold, to avoid introducing incorrect point pairs. Using a small threshold, however, causes the algorithm to converge to a local minimum, as it is unable to correct for larger displacements (e.g., when the initial transformation guess is far from the optimum). Inspired by *Trimmed ICP* [80], we sort correspondences by distance and discard those that are above the 70th percentile, under the assumption that the majority of pairs will provide sufficient information for a satisfactory alignment.

Moreover, we apply the Geman-McClure function (Fig. 3.10) on the point-to-point distance metric, like in [45], which diminishes the influence of large residuals during minimization. For a residual r , this is defined as:

$$\rho(r) = \frac{r^2/2}{\kappa + r^2}, \quad (3.33)$$

where κ is a custom scaling factor, set to 0.11 in our experiments. The optimization problem becomes:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \sum_i \rho(\|\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\|_2), \quad (3.34)$$

which can be solved using the Gauss-Newton algorithm. We parametrize $\mathbf{T}$ as a twist

3. Methodology

vector $\xi \in \mathbb{R}^6$ and we linearize around $\mathbf{I}_4$, to obtain $\mathbf{T} \approx \mathbf{I}_4 + \hat{\xi}$. The Jacobian of the residual $\mathbf{r}_i = \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i$ with respect to ξ is:

$$\mathbf{J}_{\mathbf{r}_i} = \begin{bmatrix} \mathbf{I}_3 & -\hat{\mathbf{t}} \end{bmatrix} \in \mathbb{R}^{3 \times 6}, \quad (3.35)$$

and each correspondence is associated a weight, according to the robust kernel:

$$w(\mathbf{r}_i) = \frac{\kappa^2}{(\kappa + \|\mathbf{r}_i\|_2^2)^2}. \quad (3.36)$$

The resulting normal equation is:

$$\sum_i w(\mathbf{r}_i) \mathbf{J}_{\mathbf{r}_i}^T \mathbf{J}_{\mathbf{r}_i} \xi = \sum_i w(\mathbf{r}_i) \mathbf{J}_{\mathbf{r}_i}^T \mathbf{r}_i, \quad (3.37)$$

which can be rewritten as a conventional matrix equation $\mathbf{A}\xi = \mathbf{b}$. This can be solved in several ways, but we selected the Cholesky factorization method, thanks to its numerical stability. The wanted transformation is then retrieved as $\mathbf{T} = \text{Exp}(\xi)$.

Point Cloud Filtering

We have also experimented with approaches that decrease the size of the point clouds used for registration. This is desirable because it reduces processing time and has the potential to remove points that skew registration results, such as those originating from regions occupied by trees or bushes. These regions are usually not sparse, so filtering based on the distance to the nearest neighbor or a related statistical measure was not very effective. Similarly, random downsampling is not effective, because it does not address the noise issues.

A promising method was based on the formulation described in [81], which computes a per-point entropy measure using the covariance of its neighborhood. A low entropy value indicates a structured, well-formed region (e.g., points on a planar surface), while high entropy is usually associated with noisy regions. (Fig. 3.11a) The size of the neighborhood, defined as a radius around each point, was set to 0.3m. Filtering discards points with entropy above a certain percentile, to avoid setting a fixed threshold that would not be able to adjust to variable scene features. (Fig. 3.11b) Unfortunately, this is computationally expensive and does not visibly improve registration results.

More advanced techniques involve finding salient points or structures that can be identified in multiple scans (also referred to as *landmarks*, in the SLAM paradigm), either

3. Methodology

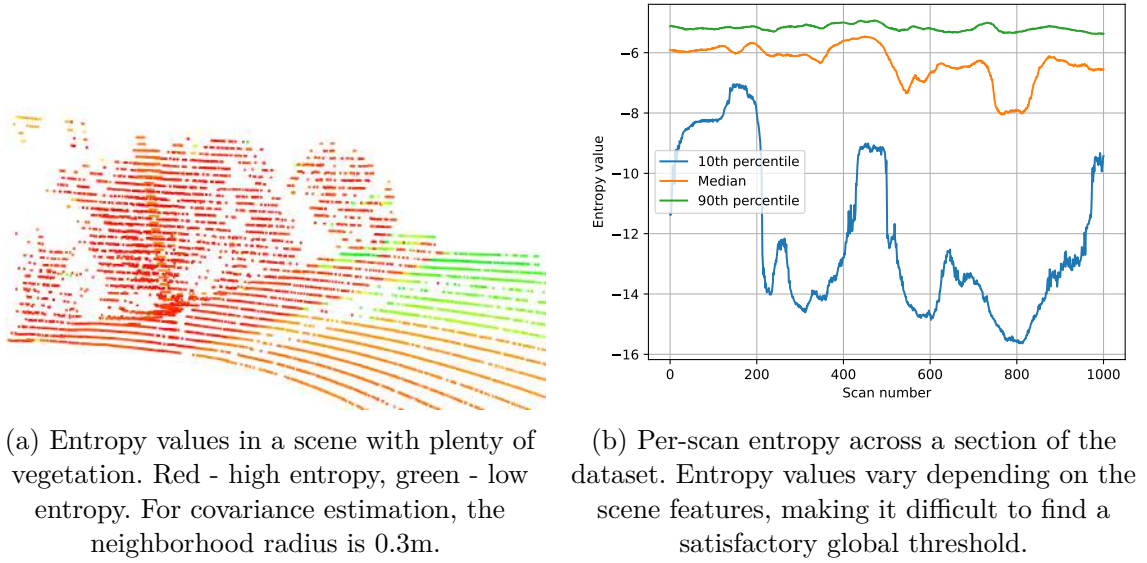


Figure 3.11: Point Cloud entropy analysis.

through feature extraction [82] or 3D segmentation, but this introduces a map representation that is not really applicable to our use case.

We employ a voxelization mechanism similar to [45], to balance computation time and noise robustness. The 3D space is divided into equal-size cells and the first point in each cell is kept. The voxel size controls the resolution of the filtered point cloud, and should be smaller than the distance threshold used for correspondence generation, to avoid degenerate scenarios.

Estimating Registration Covariance

A particular challenge is raised by the need to estimate the covariance of the pose obtained as the registration result. This serves as a confidence measure for the optimization framework, where the residuals are scaled with the inverse of $\Sigma_{\text{ICP}} \in \mathbb{R}^{6 \times 6}$. Brossard et al. [83] provided an in-depth analysis of this problem. Previous works have computed covariance based on the Jacobian involved in the correspondence distance minimization [84] or by employing a Kalman Filter that is updated for each point, using the mean squared correspondence distances as a measurement noise estimate [85].

We propose a heuristic approach based on the Root Mean Squared Error (RMSE) e of the inlier correspondences in the registration result — those for which the distance is below the provided threshold. Assuming that a perfect alignment corresponds to a

3. Methodology

covariance Σ_{ICP}^* with $\sigma_{x,y,z} = 0.01$ and $\sigma_{\phi,\theta,\psi} = 2^\circ$, we compute a factor:

$$r = 1 + \max\left(0, \frac{e - e_{\min}}{\sigma_e}\right) \quad (3.38)$$

and set $\Sigma_{\text{ICP}} = r \cdot \Sigma_{\text{ICP}}^*$. Here, σ_e represents the standard deviation of the RMSE values computed so far, so r is an indicator of how accurate the current registration is, relative to $e_{\min}$, the smallest RMSE observed up to that point. We note that RMSE can sometimes be misleading (e.g., when registration reaches a local minimum), but it provides a reasonable estimate of alignment quality in this context.

Summary

Following multiple experiments, we arrived at a procedure that can account for poor initialization and scene noise, while ensuring high-quality alignment between the current scan and the existing map. This is a two-step process, mimicking a coarse-to-fine registration procedure. Let M, S be the map and current scan point clouds, respectively, and $v(P, \alpha)$ a voxelization function that downsamples point cloud P using voxel size α (in meters). First, we perform robust point-to-point ICP between $v(S, 0.2)$ and $v(M, 0.4)$, using the motion prediction pose for initialization. The correspondence set is trimmed at the 70th percentile, and the number of iterations in the ICP procedure is not constrained, but we halt when the norm of the estimated ξ is below a small threshold (10^{-4}).

The resulting pose is used to initialize a dense registration stage: $v(S, 0.1)$ is aligned with $v(M, 0.1)$ using Generalized ICP (GICP), for at most 100 iterations. In both cases, we use a distance threshold of 0.5m, to account for corrections even in sparser areas of the point cloud (e.g., at the edges).

3.4.5 Map Handling

Because high-quality mapping constitutes one of our primary goals, having a sensible map representation is a key component of the solution. We treat the map as a point cloud in the global frame, built by concatenating a sequence of scans after transforming them with the best pose estimate computed so far. We do not attempt to store a set of key points, as this does not contribute to fine registration, but we constrain the size of the point cloud that we operate on using a sliding window approach. At every step, only the last $k = 10$ scans are considered. This straightforward method ensures temporal and

3. Methodology

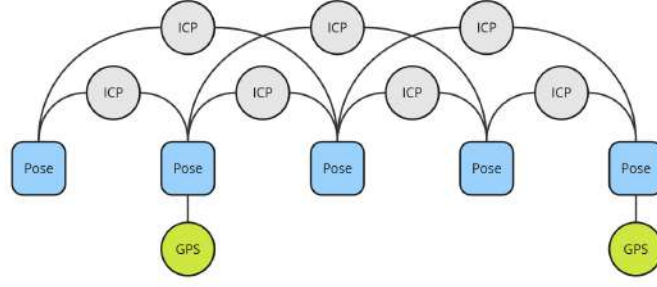


Figure 3.12: The factor graph structure.

If a GPS reading is available, a unary factor is created. The displacement predicted with point cloud registration (ICP) is used to create binary factors between consecutive poses, like a classical odometry factor. To achieve higher map quality, we also add “skip connections” based on the registration result.

spatial relevance: we do not store points that have been observed a long time in the past or points that are far away from the current location.

One drawback is that it hinders the implementation of a loop-closure mechanism, as there is no possibility to perform registration against older scans, but this is not a stringent problem, as the usage of GNSS readings prevents large localization drifts.

3.4.6 Graph Optimization

As stated earlier, we also experimented with solutions that employ factor graph optimization, thanks to its natural affinity to the SLAM problem. A factor graph [86] is a bipartite graph where nodes represent either:

- variables — these refer to variables in the problem domain; in our case, they are the estimated poses $\mathbf{T}_i$ associated with each LiDAR scan, or
- factors — probabilistic constraints, derived from measurements or prior information, whose residuals are to be minimized.

In our problem, two types of factors are employed. The GPS readings $(\mathbf{T}_i^{\text{GPS}}, \Sigma_i^{\text{GPS}})$ introduce unary factors that constrain a single pose variable. The twist vector residual is:

$$\mathbf{r}_i^{\text{GPS}}(\mathbf{T}_i) = \text{Log}\left((\mathbf{T}_i^{\text{GPS}})^{-1} \mathbf{T}_i\right). \quad (3.39)$$

Odometry measurements $(\mathbf{T}_{ij}^{\text{ICP}}, \Sigma_{ij}^{\text{ICP}})$ obtained from the point cloud registration create binary factors that are connected to the two pose variables between which the displace-

3. Methodology

ment occurred. The associated residual is:

$$\mathbf{r}_{ij}^{\text{ICP}}(\mathbf{T}_i, \mathbf{T}_j) = \text{Log} \left(\left(\mathbf{T}_{ij}^{\text{ICP}} \right)^{-1} \left(\mathbf{T}_i^{-1} \mathbf{T}_j \right) \right). \quad (3.40)$$

If we define a covariance-scaled norm $\|\mathbf{r}\|_{\Sigma}^2 = \mathbf{r}^T \Sigma^{-1} \mathbf{r}$ (note the similarity with Eq. 3.18), we can formulate the joint optimization problem:

$$\mathfrak{T}^* = \underset{\mathfrak{T}}{\text{argmin}} \left(\sum_i \left\| \mathbf{r}_i^{\text{GPS}}(\mathbf{T}_i) \right\|_{\Sigma_i^{\text{GPS}}}^2 + \sum_i \left\| \mathbf{r}_{ij}^{\text{ICP}}(\mathbf{T}_i, \mathbf{T}_j) \right\|_{\Sigma_{ij}^{\text{ICP}}}^2 \right). \quad (3.41)$$

In contrast with Section 3.4.2, this addresses the entire set of pose estimates, which is essential if trajectory corrections are to be applied retroactively and not just for the current pose.

The generic formulation of the registration pose residual $\mathbf{r}_{ij}^{\text{ICP}}$ allows for a modification that improves overall map quality. At time step k , after aligning the current scan P_k to the existing local map built from scans $\{P_{k-5}, \dots, P_{k-1}\}$, we create two odometry factors, for $(\mathbf{T}_{k-1}, \mathbf{T}_k)$ and $(\mathbf{T}_{k-2}, \mathbf{T}_k)$. The latter is motivated by the early observation that a registration-only trajectory yields a high-quality 3D map, so we use an odometry factor to enforce the registration result not only between consecutive poses. The final solution architecture, including the Factor Graph component, is presented in Fig. 3.13.

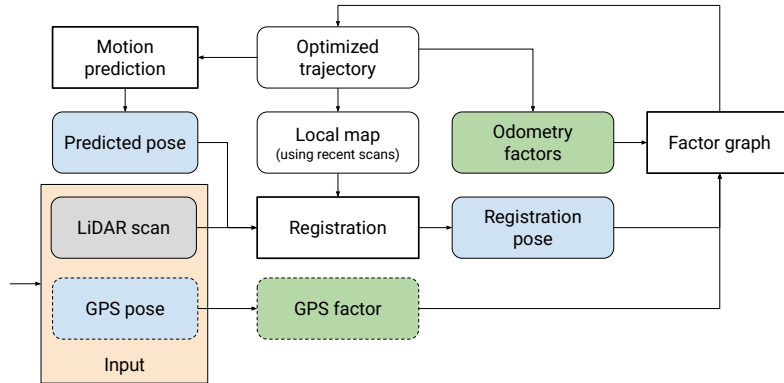


Figure 3.13: Solution architecture and workflow.

At each time step, the Factor Graph provides the best trajectory estimate using the information available so far. Registration is performed against a local map built from recent scans, using a motion-predicted pose as initial estimate. When a new pose is added to the graph, the registration result is used to create odometry factors that link it to previous poses. If a GPS pose is available, it introduces an additional factor.

3.5 Implementation details

The experiments and framework described above have been implemented in Python 3.8.10 [87], a high-level, interpreted programming language, which enabled fast development iterations. We make intensive use of existing libraries for linear algebra operations (NumPy [88], pytransform3d [89]) and point cloud processing (Open3D [90]). We treat poses and point clouds as NumPy arrays of size $(4, 4)$ and $(n, 3)$, respectively. During the registration process, the nearest-neighbor search required for correspondence generation relies on the KD-tree data structure implemented in SciPy [91]. As we do not perform any modifications to the algorithm, we use the Generalized ICP implementation provided by Open3D. For factor graph operations, we chose the popular GTSAM [92] library, despite the limited documentation available for its Python bindings.

An important mention is reserved to the Numba [93] compiler library, which enabled significant performance improvements for operations on large NumPy arrays. Performing motion compensation, for example, would not be possible in real-time (i.e., processing several LiDAR scans per second) in pure Python, but the just-in-time compilation provided by the Numba decorators, combined with parallel processing, alleviate this problem.

The code is divided into modules that group similar functionality, and we followed Object-Oriented Programming principles loosely, by having classes for the main components (e.g., Registration, Motion model, Graph optimization). For faster experimentation, the data structure presented in Section 3.3 is transformed into raw numerical form, stored on disk as `.npy` files, and re-loaded at runtime. Naturally, some custom interfacing would have to occur in order to apply the current implementation for a live robotic system, but our particular use case did not raise the need for integration with the ROS [94] framework.

Chapter 4

Results

This chapter provides a quantitative and qualitative assessment of our method, by applying it to various scenarios and analysing its output. We begin by formulating a set of odometry and mapping metrics, then observe how our solution behaves if some of its components are altered, and test it on multiple datasets.

4.1 Evaluation metrics

Let us first define the main metrics used for evaluation. In terms of trajectory, Zhang and Scaramuzza [95] provide a description of typical metrics for visual odometry, which also apply to LiDAR odometry. Given a ground-truth pose $\mathbf{T}_i = (\mathbf{R}_i, \mathbf{t}_i)$ and a corresponding pose $\widehat{\mathbf{T}}_i = (\widehat{\mathbf{R}}_i, \widehat{\mathbf{t}}_i)$, we can define translation and rotation errors as:

$$\begin{aligned} e_{\text{trans}}(\mathbf{t}_i, \widehat{\mathbf{t}}_i) &= \|\mathbf{t}_i - \widehat{\mathbf{t}}_i\|_2 \\ e_{\text{rot}}(\mathbf{R}_i, \widehat{\mathbf{R}}_i) &= \|\text{Log}(\widehat{\mathbf{R}}_i^T \mathbf{R}_i)\|_2. \end{aligned} \tag{4.1}$$

A simple way to compare two pose sequences that start at the same pose is to compute $e_{\text{trans}}(\mathbf{t}_N, \widehat{\mathbf{t}}_N)$ and $e_{\text{rot}}(\mathbf{R}_N, \widehat{\mathbf{R}}_N)$, the errors at the final state N . A more informative measure is given by the Absolute Trajectory Error (ATE) [96]:

$$\begin{aligned} \text{ATE}_{\text{trans}} &= \sqrt{\frac{1}{N} \sum_{i=1}^N e_{\text{trans}}^2(\mathbf{t}_i, \widehat{\mathbf{t}}_i)} \\ \text{ATE}_{\text{rot}} &= \sqrt{\frac{1}{N} \sum_{i=1}^N e_{\text{rot}}^2(\mathbf{R}_i, \widehat{\mathbf{R}}_i)}. \end{aligned} \tag{4.2}$$

4. Results

We compute this after transforming the ground truth and predicted trajectories such that they begin with $\mathbf{T}_0 = \mathbf{I}_4$. In the KITTI Benchmark [62], this was extended to Relative Trajectory Error (RTE), a metric that is applied on sub-sequences of a certain length or velocity, for a detailed evaluation. Given a sub-sequence (i, j) , we compute $\mathbf{t}_{i,j} = \mathbf{R}_i^T(\mathbf{t}_j - \mathbf{t}_i)$ and $\mathbf{R}_{i,j} = \mathbf{R}_i^T \mathbf{R}_j$, then derive the relative error, as a percentage, using:

$$\begin{aligned} \text{RTE}_{\text{trans}} &= \frac{e_{\text{trans}}(\mathbf{t}_{i,j}, \widehat{\mathbf{t}}_{i,j})}{\|\mathbf{t}_{i,j}\|_2} \cdot 100 \\ \text{RTE}_{\text{rot}} &= \frac{e_{\text{rot}}(\mathbf{R}_{i,j}, \widehat{\mathbf{R}}_{i,j})}{\|\text{Log}(\mathbf{R}_{i,j})\|_2} \cdot 100. \end{aligned} \quad (4.3)$$

Addressing one of the research questions that motivated this work, we also aim to quantify the quality of the resulting maps, to observe the improvements we bring to the GNSS-merged point cloud. Surprisingly, existing SLAM solutions rarely include rigorous map evaluation, because the map is seen as a localization-enabling tool, rather than a by-product of the system. The metrics here are closely related to the registration task, but differ in that they are applied on a complete 3D map, instead of a pair of point sets. We note that computing a single value for an entire point cloud is rather misleading, due to variations that appear between its regions, so we aim for point-based metrics that we can analyse through a histogram and Cumulative Distribution Function (CDF).

A high-quality 3D map is expected to be dense, consistent, and provide high detail for observed features. A very trivial metric is the nearest-neighbor distance. Without considering degenerate cases, having a small nearest-neighbor distance is desirable, but this is constrained by the resolution of the sensor — the distance would be small even when point clouds do not align well, if the LiDAR has high resolution. Alternatively, we can compute the number of points within a given radius from each point. This is a reliable measure of local density, but it is unable to distinguish between randomly-scattered points and a structured region. Inspired by the point-to-plane distance (Eq. 3.31), we can compute a metric that measures the deviation of each point from the surface determined by its neighborhood. To avoid noise, we evaluate this for regions that have a high chance of representing planar surfaces, indicated by the normal consistency c . If $\mathbf{n}$ and $\{\mathbf{n}_1, \dots, \mathbf{n}_k\}$ are the normals of a point and its neighbors, then $c = \left| \mathbf{n}^T \frac{1}{k} \sum_{i \in (1,k)} \mathbf{n}_i \right|$.

Another useful metric is given by point entropy, as described in [81]. Considering all points within a radius r around point $\mathbf{p}_k$, we compute the covariance $\Sigma(\mathbf{p}_k)$, then derive

4. Results

the differential entropy as:

$$h(\mathbf{p}_k) = \frac{1}{2} \ln (2\pi e \det (\boldsymbol{\Sigma}(\mathbf{p}_k))) \quad (4.4)$$

Adjusting the neighborhood radius controls the granularity of the metric, with a value around 0.2m providing a fair trade-off between detail level and computational cost.

To assess computational performance, we provide the processing duration (s) indicating the real time taken for a single input scan, without considering pre-processing operations, when executed on an Intel Core i7-10750H CPU.

4.2 Parameter analysis

In this section we discuss some of the main parameters of our method, in order to better understand their effect on performance, trajectory estimation and final map quality.

4.2.1 Point cloud voxelization

Voxel Size (m)	Median Duration (s)	ATE		Final		Avg. RMSE (m)	Avg. Corr.	
		tra. (m)	rot. (rad)	tra. (m)	rot. (rad)		tra. (m)	rot. (rad)
0.1	0.2984	1.2814	0.0294	2.5337	0.0467	0.0555	0.0080	0.0010
0.2	0.1684	1.3858	0.0315	2.7607	0.0511	0.0868	0.0210	0.0024
0.3	0.1586	1.5499	0.0347	3.0500	0.0542	0.1206	0.0351	0.0040
0.4	0.1589	1.7736	0.0418	3.5078	0.0659	0.1545	0.0519	0.0055
0.5	0.1603	1.4667	0.0358	2.9164	0.0568	0.1876	0.0819	0.0077
0.6	0.1635	1.9237	0.0475	3.7190	0.0677	0.2186	0.1069	0.0116

Table 4.1: Metrics for varying voxel sizes, on a sequence of 500 LiDAR scans. Using a very small voxel size ($< 0.1\text{m}$) increases computational cost and does not help with reducing point cloud noise, so we do not include that. We also report average RMSE values (after registration), alongside the average correction computed by GICP after the ICP alignment. The duration does not directly depend on the voxel size, because more ICP iterations are required for registration, when a larger voxel size is used. Overall, a smaller voxel size results in better odometry estimation and higher-quality mapping.

Following the motion compensation applied during dataset pre-processing, every input point cloud undergoes a voxelization step (as described in Section 3.4.4) designed to reduce sensor noise while keeping a sufficient amount of information for registration. We select a subsection of the trajectory consisting of 500 LiDAR scans captured over approx. 114m at 8km/h, and execute the pipeline in LiDAR-only mode (disregarding the

4. Results

GNSS information), with multiple voxel size values. As there are no GPS constraints, the graph optimization step does not modify the pose priors computed by LiDAR odometry. The corresponding GPS trajectory is treated as ground-truth, due to its low uncertainty. The results of this evaluation are presented in Table 4.1. Using a larger voxel size decreases the computational cost, as the point clouds involved have smaller resolution, but introduces considerable problems for odometry estimation, as indicated by the ATE values. We also look at the resulting maps, to confirm that a small voxel size leads to higher-quality mapping, as shown in Fig. 4.1.

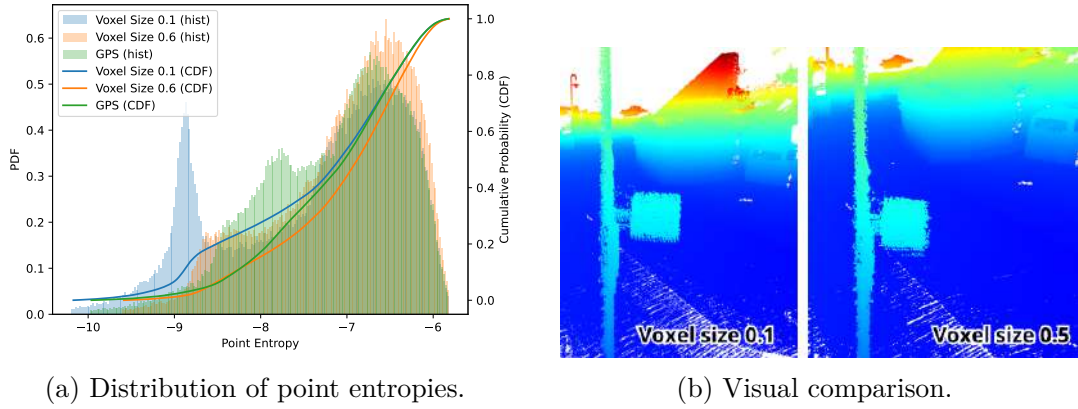


Figure 4.1: Voxel size effect on map quality: a smaller voxel size results in more accurate registration, leading to lower entropy and sharper edges.

4.2.2 Local map size

Map Size (scans)	Median Duration (s)	ATE		Final		Avg. RMSE (m)	Avg. Corr.	
		tra. (m)	rot. (rad)	tra. (m)	rot. (rad)		tra. (m)	rot. (rad)
1	0.2349	5.0058	0.1564	10.2891	0.2603	0.0940	0.0366	0.0021
2	0.2308	4.3812	0.1355	9.1949	0.2318	0.0818	0.0171	0.0016
5	0.2632	1.8325	0.0458	3.6849	0.0734	0.0615	0.0094	0.0011
10	0.2962	1.2814	0.0294	2.5337	0.0467	0.0555	0.0080	0.0010
15	0.3242	1.1645	0.0263	2.2848	0.0415	0.0540	0.0076	0.0010
25	0.3984	1.1040	0.0245	2.1504	0.0384	0.0531	0.0079	0.0011

Table 4.2: Metrics for varying map sizes, on a sequence of 500 LiDAR scans. As more past scans are used to build the local map, the registration step becomes slower, but the odometry error decreases. We use 10 previous scans, as that provides a good balance between speed and accuracy.

In this experiment, we observe the behavior of the algorithm in relation to the size of the local map, i.e., the number of previous scans that the registration is performed

4. Results

against. Given the high frequency and operational range of the sensor, consecutive scans have large overlapping areas which represent useful cues for point cloud alignment. This is proven by the low odometry errors obtained for map size ≥ 5 in Table 4.2. When larger maps are used, the computational time increases, without significant improvements for trajectory estimation.

GICP thresh. (m)	Median Duration (s)	ATE		Final		Avg. RMSE (m)	Avg. Corr.	
		tra. (m)	rot. (rad)	tra. (m)	rot. (rad)		tra. (m)	rot. (rad)
-	0.2111	1.8036	0.0371	2.8009	0.0571	0.0601	-	-
0.1	0.2949	1.1686	0.0275	1.5486	0.0357	0.0457	0.0070	0.0009
0.2	0.3020	1.1167	0.0280	1.4442	0.0340	0.0520	0.0070	0.0009
0.5	0.2860	1.1958	0.0326	1.4858	0.0390	0.0599	0.0073	0.0010

Table 4.3: Metrics for GICP variations, on a sequence of 500 LiDAR scans. We use GICP in order to improve surface alignment, as a second registration step after robust ICP. Having a larger correspondence threshold increases the risk of selecting incorrect point pairs, but helps align sparse regions of the scans.

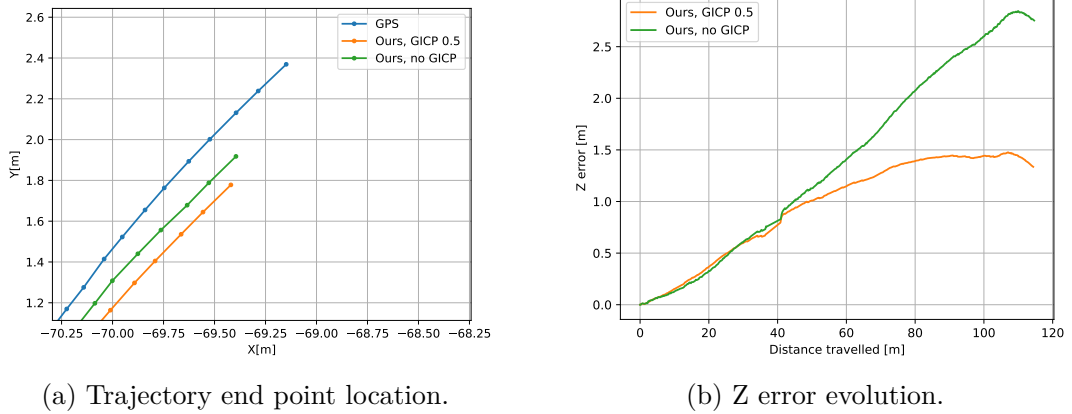


Figure 4.2: Trajectory evaluation with and without GICP. The additional GICP registration step reduces the Z-alignment error.

4.2.3 Registration strategy

We also assess the influence of the Generalized ICP registration step. During the experimental phase, we observed that this has a very positive effect on accurately matching the ground plane, and this is confirmed by the metrics in Table 4.3. For a more thorough analysis, we show the results corresponding to several distance threshold values, and compare against the case where only point-to-point ICP is used. The no-GICP version leads to higher trajectory errors overall, but we note that the Z component (elevation)

4. Results

has the highest contribution to this error. The XY error is smaller when GICP is not used (Fig. 4.2a), but the Z error is almost doubled (Fig. 4.2b). In terms of map quality, using GICP clearly helps reduce point entropies and point-to-plane errors, so the surfaces of the output map capture more details, but additional constraints (i.e., GPS) will help prevent drifts.

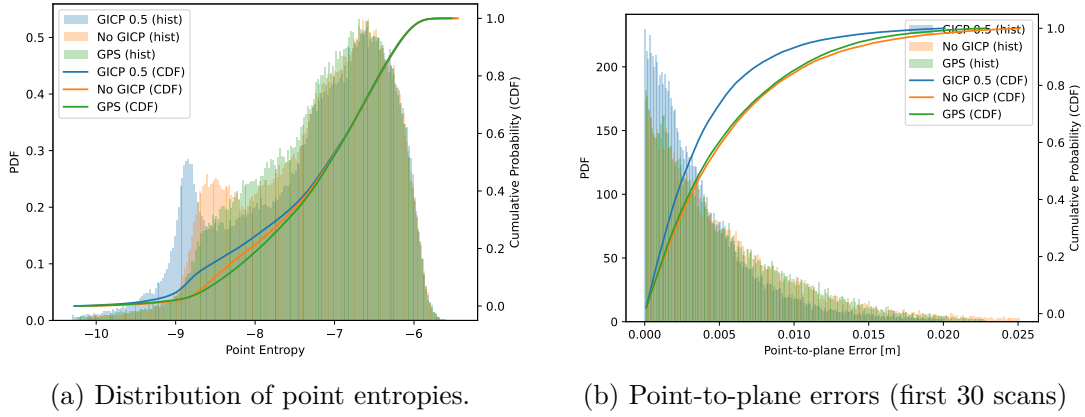


Figure 4.3: Map evaluation with and without GICP. Entropy and point-to-plane errors are lower, when GICP is used, indicating a higher-quality map.

4.3 Odometry evaluation

The first odometry evaluation we perform is on the trajectory recorded with the SDX-Compact system, depicted in Fig. 4.4. We do not use the GPS data during execution, but treat it as ground truth to obtain the trajectory error metrics in Table 4.4. Because we consider scan timestamps during motion prediction, we are able to handle large “jumps” in the scan sequence (Fig. 4.4b), something that other methods struggle with. However, elevation estimation is still an issue, as seen in Fig. 4.5b.

	ATE			Final			Avg. RTE (100m)		
	XY (m)	tra. (m)	rot. (rad)	XY (m)	tra. (m)	rot. (rad)	XY (%)	tra. (%)	rot. (%)
Full trajectory (644.4m)									
KISS-ICP	34.68	35.21	0.32	50.82	51.65	0.58	31.32	36.21	111.87
Ours	1.68	8.23	0.1	1.08	1.31	0.1	1.23	2.49	16.26
Initial section of trajectory (216m)									
KISS-ICP	0.77	4.04	0.08	1.33	4.55	0.10	0.29	4.97	38.37
Ours	1.33	5.07	0.08	2.11	8.95	0.10	0.63	2.64	24.35

Table 4.4: Comparison with KISS-ICP [45] on custom trajectory. We report average RTE for sub-sequences 100m. KISS-ICP performs better when the scans are captured at equal intervals, but diverges as soon as a significant jump occurs.

4. Results

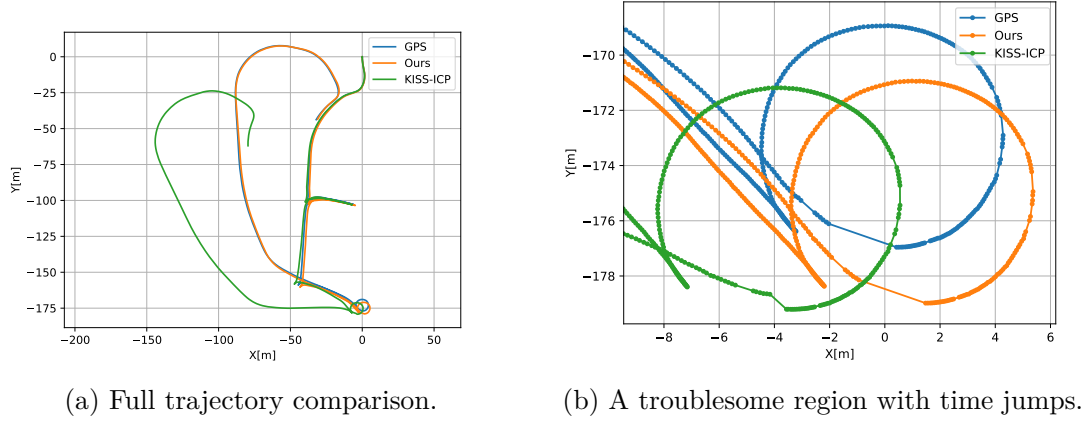


Figure 4.4: Odometry evaluation on custom trajectory: our method and KISS-ICP [45] against GPS data.

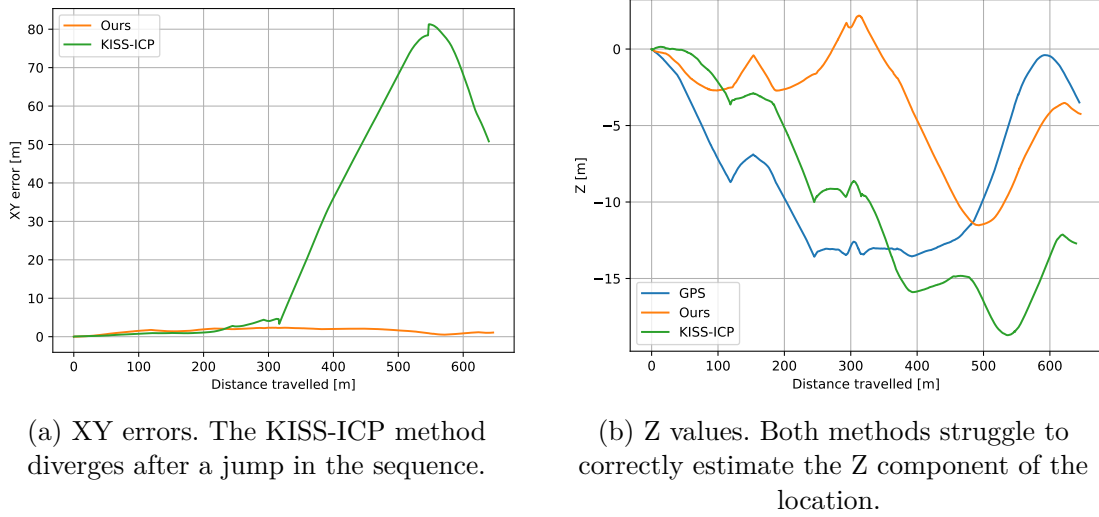


Figure 4.5: Position analysis on custom trajectory.

We also evaluate on the sequences of the KITTI dataset [97] which include ground-truth pose values from a GPS + IMU sensor system. This is a popular benchmark for urban SLAM problems, and it represents an interesting challenge for our solution because it uses a different LiDAR sensor (Velodyne HDL-64E), the environment is different, and moving vehicles are present. To operate on this dataset, we adjust the solution as follows:

- Discard LiDAR points that are farther than 50m or closer than 1m, to avoid high distortion and points belonging to the sensor rig
- Use a larger base voxel size: as the scanned environment is larger, we use a default voxel size of 0.2m instead of 0.1m

The results of this evaluation are presented in Table 4.5 and Figure 4.6.

4. Results

Seq.	Length (m)	ATE			Final			Avg. RTE (100m)		
		XY (m)	tra. (m)	rot. (rad)	XY (m)	tra. (m)	rot. (rad)	XY (%)	tra. (%)	rot. (%)
00	3723.24	7.29	16.28	0.07	10.59	11.58	0.06	0.81	1.23	12.88
01	2453.26	24.53	190.1	0.21	39.9	291.51	0.3	0.98	1.31	45.7
02	5067.02	22.2	50.39	0.13	49.72	99.53	0.22	0.7	1.22	11.99
03	560.85	2.92	8.84	0.05	6.35	18.15	0.08	0.57	0.67	7.69
04	393.65	0.72	4.98	0.03	1.16	11.76	0.06	0.39	0.89	125.81
05	2205.2	4.62	6.92	0.04	9.2	13.79	0.07	0.5	1.02	20.44
06	1232.69	2.03	3.78	0.03	4.88	8.79	0.05	0.59	0.91	54.07
07	694.39	0.5	1.8	0.02	0.96	1.36	0.02	0.58	1.0	5.12
08	3222.02	16.73	28.33	0.08	20.9	33.15	0.11	1.04	1.52	24.66
09	1704.96	6.12	14.12	0.06	9.28	9.78	0.04	0.68	1.03	8.1
10	919.4	3.94	18.8	0.07	4.61	21.78	0.09	0.65	1.04	8.79

Table 4.5: Odometry evaluation on KITTI sequences. We report the XY-only error values to emphasize localization performance. RTE values are averaged over all 100m sub-sequences. Like most dead-reckoning solutions, the ATE and Final errors increase with the length of the trajectory.

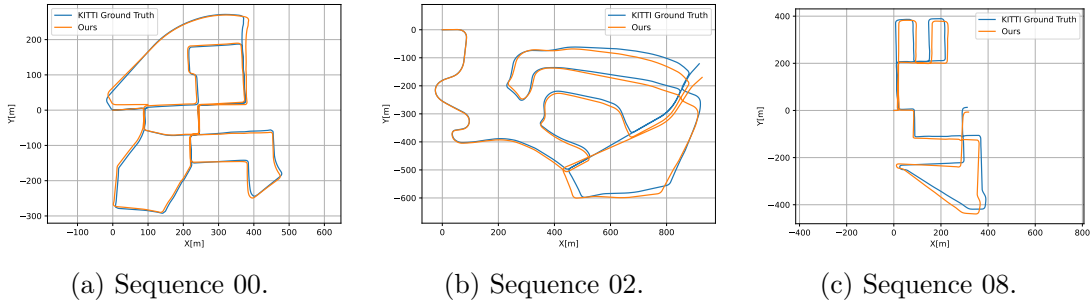


Figure 4.6: Odometry results on the three longest KITTI sequences.

4.4 GPS-integrated evaluation

So far, our method has been evaluated without any GPS input, essentially acting as a LiDAR-based dead-reckoning solution. In this section, we use the data from the GNSS receiver to imitate a real use-case, and assess the trajectory against a GPS-only approach.

First, we would like to observe how robust the solution is to GPS noise, to mimic a scenario where RTK data is not available, or the GPS signal is very poor. Starting from the existing GPS readings, we introduce artificial noise to the location values by sampling a zero-mean Gaussian distribution for each component, and modify the associated covariance based on the σ of the noise distribution. The resulting readings and covariance values are used as input for the method, simulating signal scatter. IMU readings are not affected, but we increase the covariance of the orientation components by 0.01, to allow for adjustments from the graph optimization component. We present the results of this

4. Results

experiment in Table 4.6. The noisy GPS readings are provided as input, and we evaluate the output of our method against accurate GPS readings. Beside the Absolute Trajectory Error, we compute the roughness of the obtained trajectory using the following procedure:

1. Generate subsequences of 100 consecutive scans with a sliding window approach.
2. For each subsequence, consider the discrete functions $X(t)$, $Y(t)$, $Z(t)$ such that $[X(t), Y(t), Z(t)]^T$ is the location of the ego frame at time step t .
3. Determine the polynomial functions of degree 3 that best approximate $X(t)$, $Y(t)$, $Z(t)$, and compute the “fitting error” — the mean squared error between the obtained polynomials and the original functions. A rugged trajectory will result in a higher error, as the polynomial is not able to match the non-smooth changes.
4. Average the errors across all subsequences, for each dimension.

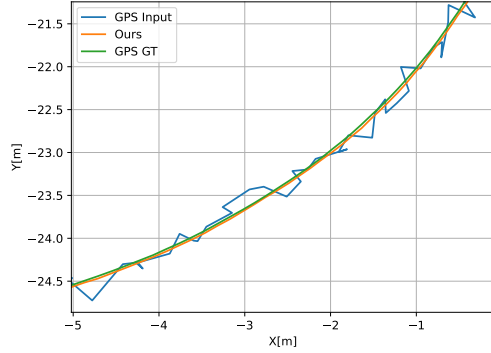
Roughness increases with the noise introduced to the GPS readings, but our method produces a smooth trajectory and reconstructs a trajectory that is close to the ground truth, effectively filtering out GPS noise. This phenomenon is shown in Figure 4.7.

A different evaluation strategy consists of dropping GPS readings at regular intervals, to simulate a lower reading frequency. We define a skip factor k and disregard the GPS information associated with LiDAR scan i , if $i \bmod k = 1$. We evaluate for multiple k values on the same sequence as before (1000 LiDAR scans, 131.7m), and present the results in Table 4.7. This is an alternative evaluation of the pure odometry capabilities, showing that our method relies very little on the GPS readings for accurate trajectory estimation, so it can be applied to systems that do not have a high-frequency GPS receiver. We observe that this is equivalent to having regions where the GPS fix is lost and re-acquired after a period of time, indicating that our method is also capable of handling such scenarios, common when the vehicle is traversing a vegetation-filled region.

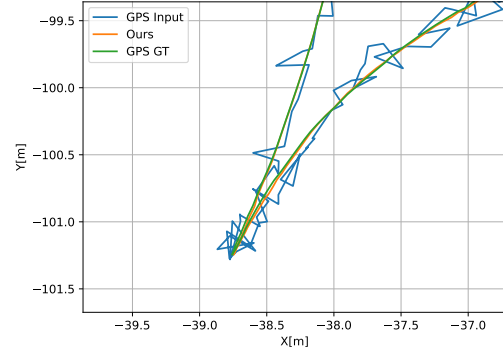
4. Results

Noise σ (m)	Noisy GPS Input					Ours				
	ATE (m)		Roughness			ATE (m)		Roughness		
	XY	tra.	X	Y	Z	XY	tra.	X	Y	Z
0.05	0.114	0.134	0.005	0.007	0.002	0.029	0.032	0.003	0.005	0.000
0.1	0.228	0.268	0.011	0.014	0.009	0.058	0.067	0.003	0.005	0.001
0.5	1.140	1.340	0.218	0.241	0.228	0.285	0.571	0.015	0.018	0.013
1.0	2.280	2.680	0.866	0.947	0.910	0.707	1.219	0.058	0.064	0.056
2.0	4.559	5.359	3.457	3.768	3.641	2.165	7.623	0.2	0.212	0.223

Table 4.6: Trajectory evaluation with noisy GPS input, on a sequence of 1000 LiDAR scans (approx. 131.7m), against the zero-noise GPS readings. Roughness is measured by fitting a cubic polynomial to subsequences of each dimension — X, Y and Z, as a function of time — and averaging the fitting error across all subsequences. Our method filters out GPS noise, reconstructing a trajectory that is smoother and closer to the ground truth.



(a) A curved region.



(b) A corner region.

Figure 4.7: Trajectory corrections on noisy GPS input, $\sigma = 0.1$. The path computed by our method (orange) is smoother and closer to the expected path (green) compared to the noisy GPS path (blue).

GPS Skip Factor	GPS Count	ATE			Final		
		XY (m)	translation (m)	rotation (rad)	XY (m)	translation (m)	rotation (rad)
—	1000	0.003	0.026	0.001	0.005	0.040	0.001
5	200	0.009	0.021	0.001	0.008	0.037	0.002
10	100	0.012	0.016	0.002	0.010	0.027	0.002
50	20	0.022	0.025	0.005	0.026	0.028	0.007
100	10	0.027	0.034	0.007	0.050	0.055	0.012
334	3	0.070	0.121	0.013	0.036	0.146	0.019
500	2	0.107	0.445	0.019	0.107	0.741	0.028
1000	1	1.173	3.869	0.067	1.543	6.083	0.090

Table 4.7: Trajectory evaluation with varying GPS Skip Factor, on a sequence of 1000 LiDAR scans. The ground truth that we test against are the GPS readings provided as input to the method. As expected, the error increases as GPS frequency decreases, but the magnitude of the errors remains small, even when few readings are used.

4.5 Mapping evaluation

We are interested in observing how our solution affects map quality, in comparison with the 3D map constructed by simply pasting the LiDAR scans at the RTK-corrected poses. Every scan is transformed according to the corresponding pose in the trajectory, to obtain global point coordinates. Table 4.8 provides a preliminary visual evaluation on KITTI data, with very clear improvements: edges are sharper than when using only GPS measurements. The formal evaluation consists of plotting the distribution of the point entropy values (see Eq. 4.4) and point-to-plane error in various scenes, as shown in Table 4.9. We compute point-to-plane error as the distance between a point and a plane fit to its local neighborhood ($k = 10$ nearest neighbors, with distance threshold $d = 0.2\text{m}$), for points that belong to planar patches. Planar patches are identified using the `detect_planar_patches` function of the Open3D library⁶. To obtain high-quality patches, we use a maximum allowed ratio of outliers of 0.05 (`outlier_ratio`) and constrain the variance of the point normals below 5 degrees (`normal_variance_threshold_deg`). We also plot the output obtained when only every 10th GPS reading is used — fewer GPS constraints enable a higher contribution of the registration result to the final pose estimation. A similar effect can be achieved by increasing the covariance of the INS reading. The results indicate that this improves map quality without damaging the accuracy of the trajectory, particularly in more structured scenes, or where the GPS signal is inaccurate.

⁶Open3D Documentation for `detect_planar_patches`

4. Results

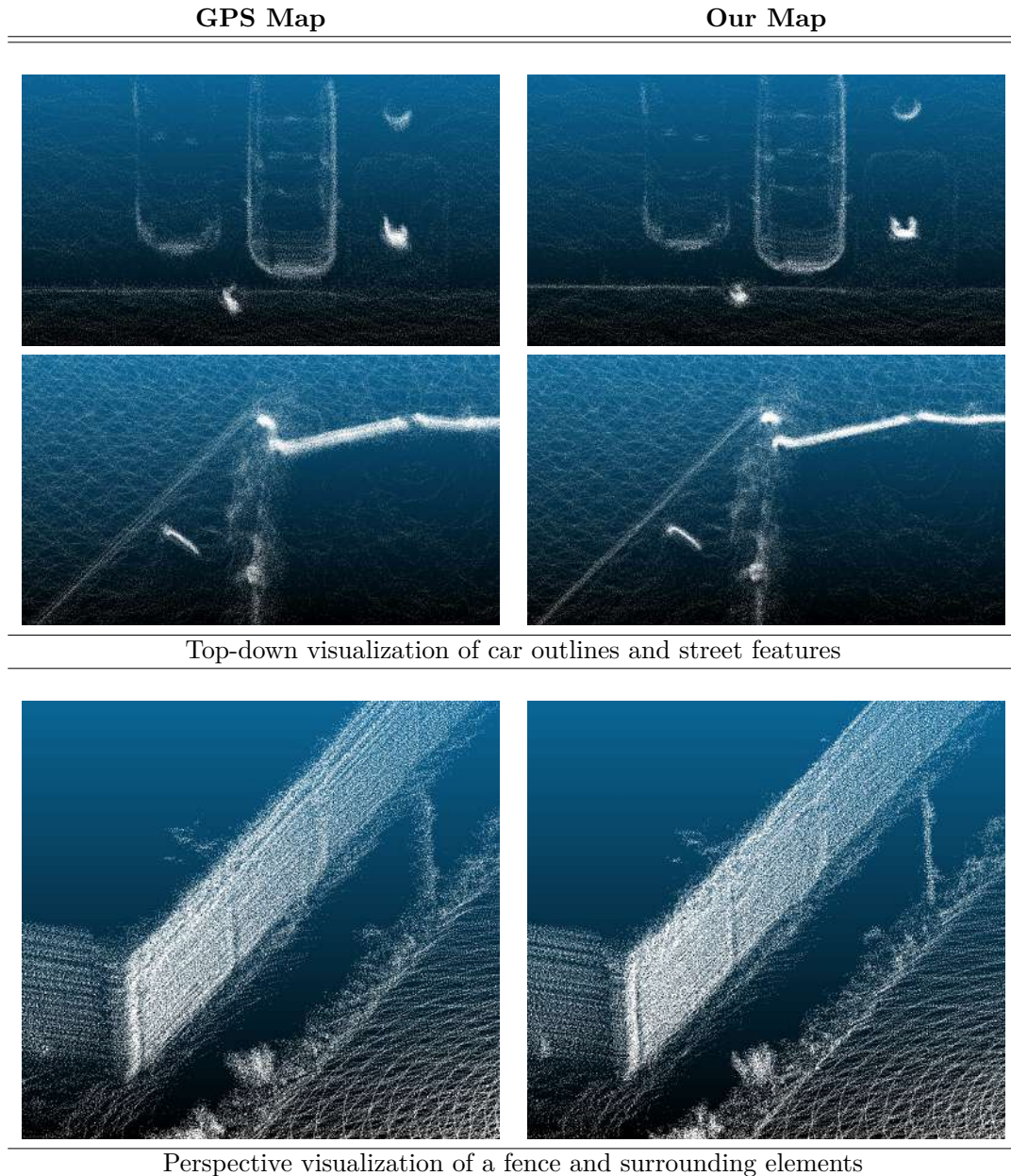


Table 4.8: Visual map comparison on KITTI sequence 04. The GPS map is constructed by transforming the LiDAR scans according to the ground truth GPS poses, while our map uses the graph-optimized poses. In our map, contours and scene details are visibly sharper, indicating better scan alignment. The visualization was generated using CloudCompare.

4. Results

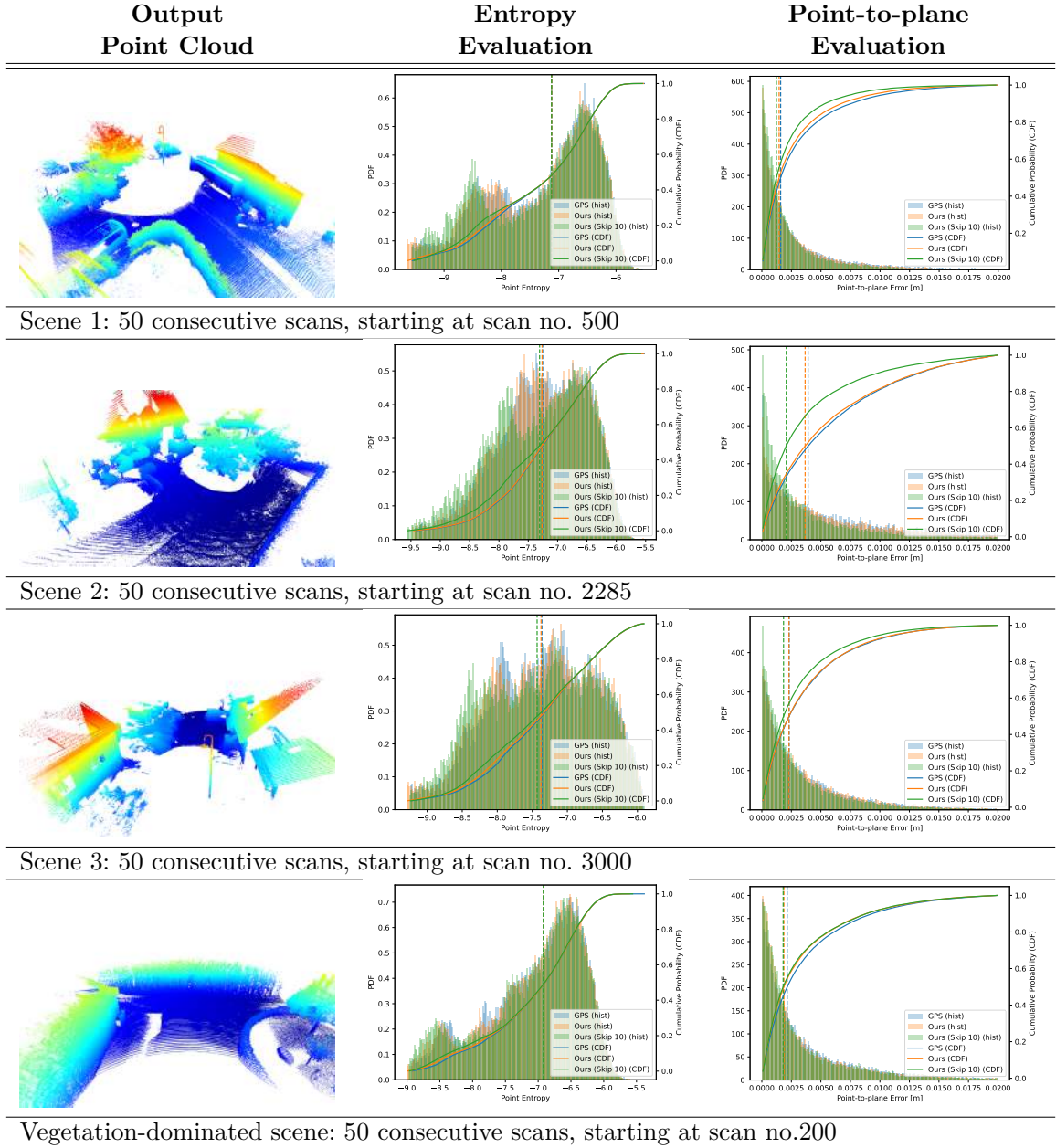


Table 4.9: Map evaluation in different scenes of our original dataset, comparing raw GPS (blue), our approach with full GPS constraints (orange) and our approach with every 10th GPS reading (green). Point clouds are colored by height, for visualization.

For each distribution, the dashed vertical line indicates the median value.

Compared to GPS-only mapping, our method reduces point entropy and point-to-plane errors, which is equivalent to having smoother planar surfaces and finer visible details.

Using every 10th GPS reading improves the map quality even further, as the model relies more on scan alignment.

Chapter 5

Conclusion

This work addressed odometry, a fundamental topic of mobile robotics, with emphasis on the area of perception, and was driven by its applicability to an industrial product, the SDX-Compact manufactured by Sodex Innovations GmbH. Range information from a 360-degree LiDAR sensor is fused with localization and orientation measurements provided by an INS, to increase robustness to GPS perturbations and enable 3D mapping quality beyond the accuracy of RTK-corrected readings.

We provided a review of the current state of research around this problem and its many facets. This was followed by a presentation of the hardware and data that we operate on, and a detailed explanation of the experiments conducted around each component of our final solution. Our contribution is an original displacement estimation method which relies on a two-step scan alignment process (robust ICP and Generalized ICP) and a factor graph structure, where registration results are combined with absolute GPS readings. To enforce point cloud registration constraints, we add “skip connections” between non-consecutive poses, based on the ICP-estimated displacement. GPS readings introduce unary factors which prevent trajectory drift and remove the need for a loop closure mechanism. The results show that our method is able to handle LiDAR scans captured at irregular time intervals, noisy GPS, or gaps in the sequence of GPS measurements.

We test our solution in multiple scenarios, with varying parameter configurations, and evaluate odometry on the KITTI dataset [97] and on a custom dataset collected with the SDX-Compact. Unlike other odometry or SLAM approaches, we are also interested in the mapping quality — the complete point cloud captured along the trajectory — so we evaluate the output 3D map against GPS-only results. We obtain smoother and more fine-detailed maps, thanks to the registration constraints.

5. Conclusion

Guided by the research questions laid out in the initial stage of the project, the conducted work included multiple experiments which allow us to provide concrete and detailed answers.

1. *What metrics exist for measuring the accuracy of point cloud registration?*

Point cloud registration can be evaluated in several ways, depending on the use case and the desired result. If ground truth displacement is available (e.g., from an external sensor system or by designing a scene with custom objects which can be identified and localized precisely), it can be compared to the transformation computed by the registration algorithm. In most scenarios, however, registration accuracy is approximated as an error function between the two scans, after they have been transformed to a common reference frame. The metrics are a combination of distance measurements between 3D elements identified in both point clouds (correspondences), such as points, planes, edges, etc., aiming to quantify the geometric consistency of the combined point cloud. Sensor noise and outliers introduce particular challenges, if a generic metric is sought. In our case, registration accuracy is indicated by the RMSE value obtained after applying GICP, which is a point-to-point metric. Alternative metrics can be formulated using covariance estimation for local 3D neighborhoods [81] or the transformation resulting from the registration process [98].

2. *Can methods that use only visual information achieve higher quality point cloud registration (3D mapping) than merging based on RTK?*

Registering point clouds using the intrinsic visual information generally results in higher quality 3D maps — we have obtained maps that have lower entropy and smoother surfaces. However, this is subject to having an accurate motion estimation mechanism, in order to perform scan de-skewing and ensure that the range information from the LiDAR sensor is representative of the real world. Because this depends on the velocity of the vehicle and the features present in the environment, it could be argued that the LiDAR output alone is not sufficient for accurate registration.

3. *To what extent is LiDAR-based odometry an alternative to GNSS localization?*

We have designed, implemented and evaluated a LiDAR-based odometry method which achieves, on average, less than 1m of drift in the horizontal plane, when travelling for 100m (according to the results in Table 4.5). Unsurprisingly, this highlights a general limitation of dead-reckoning methods: small errors that accumulate over

5. Conclusion

time, resulting in poor localization. Therefore, employing LiDAR-based odometry as an alternative to GNSS localization depends on the use-case. If it is used to approximate relatively small local displacements (e.g., to increase the frequency of state estimation beyond the frequency of the GPS receiver) or if the the GNSS readings have large uncertainty, LiDAR odometry proves a reliable solution. Another limitation is the nature of the environment, as point cloud registration cannot be accurately performed in degenerate or featureless scenes.

5.1 Future Work

There are several directions worth investigating for improving the current approach, if more time and human resource is available. One particular issue that surfaced during the evaluation stage was the high drift in the vertical axis. Although it is not unexpected, because we did not employ any particular methods for preventing this (unlike LeGO-LOAM [51], for example), a thorough investigation is required. It could be hypothesized that this is caused by the GICP registration overfitting the alignment between ground regions of the point clouds, in the presence of noisy outliers, or some inaccuracy in the scan de-skewing process.

Performing motion compensation without relying on the poses provided by the INS is another potential improvement which would make our solution applicable to a wider range of systems. This is a vast research problem that could be tackled by extracting the raw IMU measurements from the available sensor, or by developing a dedicated component for correcting the incoming LiDAR scans.

Another limitation of our method is that it does not take into account the beam intensity values provided by the sensor or the RGB camera output, which could prove very useful for registering scans in areas with few useful 3D features. There are methods for performing ICP on point clouds with associated color information [99], and LiDAR odometry has benefited from intensity data in the past [52].

Although performance was not one of our immediate constraints, we believe that the computational cost of our approach could be reduced drastically if the implementation were done in C++ instead of Python, to meet the requirements of real-time systems. Creating a ROS 2 interface that enables arbitrary robotic setups to forward LiDAR and GPS data to our solution is another possible improvement on the software side.

Bibliography

- [1] Eric Roberts. *Robotics: A Brief History*. <https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/robotics/history.html>. Accessed: 2025-02-01. 1999.
- [2] Mario Taddei. *Leonardo's Robots*. <https://tinyurl.com/mt2a4s9x>. Archived version accessed: 2025-02-01. 2008.
- [3] Boston Dynamics. *Leaps, Bounds, and Backflips*. <https://bostondynamics.com/blog/leaps-bounds-and-backflips/>. Accessed: 2025-02-01. 2023.
- [4] Subhdeep Mukherjee et al. "Humanoid robot in healthcare: A Systematic Review and Future Research Directions". In: *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*. Vol. 1. 2022, pp. 822–826. DOI: 10.1109/COM-IT-CON54601.2022.9850577.
- [5] Kai-Yi Chin, Chin-Hsien Wu, and Zeng-Wei Hong. "A Humanoid Robot as a Teaching Assistant for Primary Education". In: *2011 Fifth International Conference on Genetic and Evolutionary Computing*. 2011, pp. 21–24. DOI: 10.1109/ICGEC.2011.13.
- [6] Uvais Qidwai, Saad Bin Abul Kashem, and Olcay Conor. "Humanoid Robot as a Teacher's Assistant: Helping Children with Autism to Learn Social and Academic Skills". In: *Journal of Intelligent & Robotic Systems* 98.3 (2020), pp. 759–770. ISSN: 1573-0409. DOI: 10.1007/s10846-019-01075-1. URL: <https://doi.org/10.1007/s10846-019-01075-1>.
- [7] Do Hyoung Kim et al. "Development of a Facial Expression Imitation System". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006, pp. 3107–3112. DOI: 10.1109/IR0S.2006.282329.
- [8] Francis B Colavita. "Human sensory dominance". In: *Perception & Psychophysics* 16.2 (1974), pp. 409–412.

BIBLIOGRAPHY

- [9] Fabian Huttmacher. “Why Is There So Much More Research on Vision Than on Any Other Sensory Modality?” In: *Frontiers in Psychology* 10 (2019), p. 2246. DOI: 10.3389/fpsyg.2019.02246. URL: <https://doi.org/10.3389/fpsyg.2019.02246>.
- [10] David Hecht and Miriam Reiner. “Sensory dominance in combinations of audio, visual and haptic stimuli”. In: *Experimental brain research* 193 (2009), pp. 307–314.
- [11] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.
- [12] Josep Aulinas et al. “The SLAM problem: a survey”. In: *Artificial Intelligence Research and Development* (2008), pp. 363–371.
- [13] Rudolph Emil Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.
- [14] A.J. Davison and D.W. Murray. “Simultaneous localization and map-building using active vision”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), pp. 865–880. DOI: 10.1109/TPAMI.2002.1017615.
- [15] Chunbo Luo et al. “UAV position estimation and collision avoidance using the extended Kalman filter”. In: *IEEE Transactions on vehicular technology* 62.6 (2013), pp. 2749–2762.
- [16] Albert Palomer, Pere Ridao, and David Ribas. “Inspection of an underwater structure using point-cloud SLAM with an AUV and a laser scanner”. In: *Journal of field robotics* 36.8 (2019), pp. 1333–1344.
- [17] Pierre Del Moral. “Nonlinear filtering: Interacting particle resolution”. In: *Comptes Rendus de l’Académie des Sciences-Series I-Mathematics* 325.6 (1997), pp. 653–658.
- [18] Fuyu Nie et al. “LCPF: A Particle Filter Lidar SLAM System With Loop Detection and Correction”. In: *IEEE Access* 8 (2020), pp. 20401–20412. DOI: 10.1109/ACCESS.2020.2968353.
- [19] Hans Peter Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover*. Stanford University, 1980.
- [20] Andrew J. Davison et al. “MonoSLAM: Real-Time Single Camera SLAM”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 1052–1067. DOI: 10.1109/TPAMI.2007.1049.

BIBLIOGRAPHY

- [21] Georg Klein and David Murray. “Parallel tracking and mapping for small AR workspaces”. In: *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE. 2007, pp. 225–234.
- [22] Christopher Mei et al. “RSLAM: A system for large-scale mapping in constant-time using stereo”. In: *International journal of computer vision* 94 (2011), pp. 198–214.
- [23] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-Squares Fitting of Two 3-D Point Sets”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), pp. 698–700. DOI: 10.1109/TPAMI.1987.4767965.
- [24] Paul J Besl and Neil D McKay. “Method for registration of 3-D shapes”. In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606.
- [25] Szymon Rusinkiewicz and Marc Levoy. “Efficient variants of the ICP algorithm”. In: *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE. 2001, pp. 145–152.
- [26] Xiaoshui Huang et al. *A comprehensive survey on point cloud registration*. 2021. arXiv: 2103.02690 [cs.CV]. URL: <https://arxiv.org/abs/2103.02690>.
- [27] Y. Chen and G. Medioni. “Object modeling by registration of multiple range images”. In: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. 1991, 2724–2729 vol.3. DOI: 10.1109/ROBOT.1991.132043.
- [28] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. “Generalized-ICP”. In: *Robotics: science and systems*. Vol. 2. 4. Seattle, WA. 2009, p. 435.
- [29] T. Masuda, K. Sakaue, and N. Yokoya. “Registration and integration of multiple range images for 3-D model construction”. In: *Proceedings of 13th International Conference on Pattern Recognition*. Vol. 1. 1996, 879–883 vol.1. DOI: 10.1109/ICPR.1996.546150.
- [30] Greg Turk and Marc Levoy. “Zippered polygon meshes from range images”. In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 1994, pp. 311–318.
- [31] Guy Godin, Marc Rioux, and Rejean Baribeau. “Three-dimensional registration using range and intensity information”. In: *Videometrics III*. Vol. 2350. SPIE. 1994, pp. 279–290.

BIBLIOGRAPHY

- [32] Kari Pulli. “Multiview registration for large data sets”. In: *Second international conference on 3-d digital imaging and modeling (cat. no. pr00062)*. IEEE. 1999, pp. 160–168.
- [33] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. “Sparse iterative closest point”. In: *Computer graphics forum*. Vol. 32. 5. Wiley Online Library. 2013, pp. 113–123.
- [34] Gérard Blais and Martin D. Levine. “Registering multiview range data to create 3D computer objects”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 820–824.
- [35] Peter Biber and Wolfgang Straßer. “The normal distributions transform: A new approach to laser scan matching”. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*. Vol. 3. IEEE. 2003, pp. 2743–2748.
- [36] Martin Magnusson, Achim Lilienthal, and Tom Duckett. “Scan registration for autonomous mining vehicles using 3D-NDT”. In: *Journal of Field Robotics* 24.10 (2007), pp. 803–827.
- [37] Martin Magnusson et al. “Evaluation of 3D registration reliability and speed-A comparison of ICP and NDT”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 3907–3912.
- [38] Yueqian Shen et al. “MI-NDT: Multiscale Iterative Normal Distribution Transform for Registering Large-Scale Outdoor Scans”. In: *IEEE Transactions on Geoscience and Remote Sensing* 62 (2024), pp. 1–13. DOI: 10.1109/TGRS.2024.3437162.
- [39] Shoubin Chen et al. “NDT-LOAM: A real-time LiDAR odometry and mapping with weighted NDT and LFA”. In: *IEEE Sensors Journal* 22.4 (2021), pp. 3660–3671.
- [40] Zan Gojcic et al. “The perfect match: 3d point cloud matching with smoothed densities”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5545–5554.
- [41] Haowen Deng, Tolga Birdal, and Slobodan Ilic. “Ppfnet: Global context aware local features for robust 3d point matching”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 195–205.

BIBLIOGRAPHY

- [42] Alireza Asvadi et al. “3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes”. In: *Robotics and Autonomous Systems* 83 (2016), pp. 299–311.
- [43] Liang Chen, Jian Yang, and Hui Kong. “Lidar-histogram for fast road and obstacle detection”. In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 1343–1348.
- [44] Ji Zhang, Sanjiv Singh, et al. “LOAM: Lidar odometry and mapping in real-time.” In: *Robotics: Science and systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.
- [45] Ignacio Vizzo et al. “KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way”. In: *IEEE Robotics and Automation Letters (RA-L)* 8.2 (2023), pp. 1029–1036. DOI: 10.1109/LRA.2023.3236571.
- [46] Pierre Dellenbach et al. “CT-ICP: Real-time Elastic LiDAR Odometry with Loop Closure”. In: *CoRR* abs/2109.12979 (2021). arXiv: 2109.12979. URL: <https://arxiv.org/abs/2109.12979>.
- [47] Yue Pan et al. “MULLS: Versatile LiDAR SLAM via Multi-metric Linear Least Square”. In: *CoRR* abs/2102.03771 (2021). arXiv: 2102.03771. URL: <https://arxiv.org/abs/2102.03771>.
- [48] Kenny Chen et al. “Direct LiDAR Odometry: Fast Localization With Dense Point Clouds”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 2000–2007. DOI: 10.1109/LRA.2022.3142739.
- [49] Daehan Lee, Hyungtae Lim, and Soohye Han. “GenZ-ICP: Generalizable and Degeneracy-Robust LiDAR Odometry Using an Adaptive Weighting”. In: *IEEE Robotics and Automation Letters* (2024).
- [50] Kenji Koide et al. “Glim: 3d range-inertial localization and mapping with gpu-accelerated scan matching factors”. In: *Robotics and Autonomous Systems* 179 (2024), p. 104750.
- [51] Tixiao Shan and Brendan Englot. “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4758–4765.

BIBLIOGRAPHY

- [52] Patrick Pfreundschuh et al. “COIN-LIO: Complementary Intensity-Augmented LiDAR Inertial Odometry”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 1730–1737.
- [53] Tixiao Shan et al. “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping”. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5135–5142.
- [54] Wei Xu and Fu Zhang. “FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324. DOI: 10.1109/LRA.2021.3064227.
- [55] Deyu Yin et al. “CAE-LO: LiDAR Odometry Leveraging Fully Unsupervised Convolutional Auto-Encoder for Interest Point Detection and Feature Description”. In: *CoRR* abs/2001.01354 (2020). arXiv: 2001.01354. URL: <http://arxiv.org/abs/2001.01354>.
- [56] Julian Nubert, Shehryar Khattak, and Marco Hutter. “Self-supervised learning of lidar odometry for robotic applications”. In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2021, pp. 9601–9607.
- [57] Zhichao Li and Naiyan Wang. “Dmlo: Deep matching lidar odometry”. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 6010–6017.
- [58] Qing Li et al. “Lo-net: Deep real-time lidar odometry”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8473–8482.
- [59] Hongliang Guo, Jiankang Zhu, and Yunping Chen. “E-LOAM: LiDAR odometry and mapping with expanded local structural information”. In: *IEEE transactions on intelligent vehicles* 8.2 (2022), pp. 1911–1921.
- [60] Kenneth Levenberg. “A method for the solution of certain non-linear problems in least squares”. In: *Quarterly of applied mathematics* 2.2 (1944), pp. 164–168.
- [61] Donald W. Marquardt. “An Algorithm for Least-Squares Estimation of Nonlinear Parameters”. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. DOI: 10.1137/0111030. eprint: <https://doi.org/10.1137/0111030>. URL: <https://doi.org/10.1137/0111030>.

BIBLIOGRAPHY

- [62] Andreas Geiger, Philip Lenz, and Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3354–3361.
- [63] Ji Zhang and Sanjiv Singh. “Visual-lidar odometry and mapping: Low-drift, robust, and fast”. In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2015, pp. 2174–2181.
- [64] Han Wang et al. “F-loam: Fast lidar odometry and mapping”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 4390–4396.
- [65] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.
- [66] Hesai Technology. *XT16/32/32M High-Precision 360° Mid-Range Lidar*. <https://www.hesaitech.com/product/xt16-32-32m/>. Accessed: 2025-02-28.
- [67] Tiziano Guadagnino et al. *Kinematic-ICP: Enhancing LiDAR Odometry with Kinematic Constraints for Wheeled Mobile Robots Moving on Planar Surfaces*. 2025. arXiv: 2410.10277 [cs.R0]. URL: <https://arxiv.org/abs/2410.10277>.
- [68] Septentrio. *AsterX SBI3 Pro+*. Accessed: 2025-02-28. 2025. URL: <https://www.septentrio.com/en/products/gnss-receivers/gnss-ins-ruggedized-boxes/asterx-sbi3-pro-plus>.
- [69] Septentrio. *FUSE+ Sensor Fusion*. Accessed: 2025-02-28. 2025. URL: <https://www.septentrio.com/en/learn-more/advanced-positioning-technology/fuse-sensor-fusion>.
- [70] Xin Nie et al. “Two-Step Self-Calibration of LiDAR-GPS/IMU Based on Hand-Eye Method”. In: *Symmetry* 15.2 (2023). ISSN: 2073-8994. DOI: 10.3390/sym15020254. URL: <https://www.mdpi.com/2073-8994/15/2/254>.
- [71] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *CoRR* abs/1903.11027 (2019). arXiv: 1903.11027. URL: <http://arxiv.org/abs/1903.11027>.
- [72] Jean-Luc Déziel et al. “PixSet : An Opportunity for 3D Computer Vision to Go Beyond Point Clouds With a Full-Waveform LiDAR Dataset”. In: *CoRR* abs/2102.12010 (2021). arXiv: 2102.12010. URL: <https://arxiv.org/abs/2102.12010>.

BIBLIOGRAPHY

- [73] Matthew McDermott and Jason Rife. “Correcting Motion Distortion for LIDAR Scan-to-Map Registration”. In: *IEEE Robotics and Automation Letters* 9.2 (2024), pp. 1516–1523. DOI: 10.1109/LRA.2023.3346757.
- [74] Seungpyo Hong, Heedong Ko, and Jinwook Kim. “VICP: Velocity updating iterative closest point algorithm”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 1893–1898. DOI: 10.1109/ROBOT.2010.5509312.
- [75] Lei He, Zhe Jin, and Zhenhai Gao. “De-Skewing LiDAR Scan for Refinement of Local Mapping”. In: *Sensors* 20.7 (2020). ISSN: 1424-8220. DOI: 10.3390/s20071846. URL: <https://www.mdpi.com/1424-8220/20/7/1846>.
- [76] Iulian Filip et al. “Lidar SLAM Comparison in a Featureless Tunnel Environment”. In: *2022 22nd International Conference on Control, Automation and Systems (ICCAS)*. 2022, pp. 1648–1653. DOI: 10.23919/ICCAS55662.2022.10003820.
- [77] Frank Dellaert, Michael Kaess, et al. “Factor graphs for robot perception”. In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.
- [78] Linfei Pan et al. “Global structure-from-motion revisited”. In: *European Conference on Computer Vision*. Springer. 2024, pp. 58–77.
- [79] Richard Hartley et al. “Rotation averaging”. In: *International journal of computer vision* 103 (2013), pp. 267–305.
- [80] D. Chetverikov et al. “The Trimmed Iterative Closest Point algorithm”. In: *2002 International Conference on Pattern Recognition*. Vol. 3. 2002, 545–548 vol.3. DOI: 10.1109/ICPR.2002.1047997.
- [81] Daniel Adolfosson et al. “Coral—are the point clouds correctly aligned?” In: *2021 European conference on mobile robots (ECMR)*. IEEE. 2021, pp. 1–7.
- [82] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. “Fast Point Feature Histograms (FPFH) for 3D registration”. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 3212–3217. DOI: 10.1109/ROBOT.2009.5152473.
- [83] Martin Brossard, Silvere Bonnabel, and Axel Barrau. “A New Approach to 3D ICP Covariance Estimation”. In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 744–751. ISSN: 2377-3774. DOI: 10.1109/lra.2020.2965391. URL: <http://dx.doi.org/10.1109/LRA.2020.2965391>.

BIBLIOGRAPHY

- [84] Sai Manoj Prakhya et al. “A closed-form estimate of 3D ICP covariance”. In: *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*. 2015, pp. 526–529. DOI: 10.1109/MVA.2015.7153246.
- [85] Rick H. Yuan, Clark N. Taylor, and Scott L. Nykl. “Accurate Covariance Estimation for Pose Data From Iterative Closest Point Algorithm”. In: *NAVIGATION: Journal of the Institute of Navigation* 70.2 (2023), navi.562. ISSN: 0028-1522. DOI: 10.33012/navi.562. eprint: <https://navi.ion.org/content/70/2/navi.562.full.pdf>. URL: <https://navi.ion.org/content/70/2/navi.562>.
- [86] Frank Dellaert. “Factor graphs and GTSAM: A hands-on introduction”. In: *Georgia Institute of Technology, Tech. Rep* 2.4 (2012).
- [87] Guido van Rossum and Python Software Foundation. *Python Programming Language*. Version 3.8.10. 2025. URL: <https://www.python.org>.
- [88] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- [89] Alexander Fabisch. “pytransform3d: 3D Transformations for Python”. In: *Journal of Open Source Software* 4.33 (2019), p. 1159. DOI: 10.21105/joss.01159. URL: <https://doi.org/10.21105/joss.01159>.
- [90] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).
- [91] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.
- [92] Frank Dellaert and GTSAM Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. DOI: 10.5281/zenodo.5794541. URL: <https://github.com/borglab/gtsam>.
- [93] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. “Numba: a LLVM-based Python JIT compiler”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM ’15. Austin, Texas: Association for Computing Machinery, 2015. ISBN: 9781450340052. DOI: 10.1145/2833157.2833162. URL: <https://doi.org/10.1145/2833157.2833162>.
- [94] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: vol. 3. Jan. 2009.

BIBLIOGRAPHY

- [95] Zichao Zhang and Davide Scaramuzza. “A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7244–7251.
- [96] Rainer Kümmerle et al. “On measuring the accuracy of SLAM algorithms”. In: *Autonomous Robots* 27 (2009), pp. 387–407.
- [97] Andreas Geiger et al. “Vision meets robotics: The kitti dataset”. In: *The international journal of robotics research* 32.11 (2013), pp. 1231–1237.
- [98] Andrea Censi. “An accurate closed-form estimate of ICP’s covariance”. In: *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE. 2007, pp. 3167–3172.
- [99] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. “Colored point cloud registration revisited”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 143–152.

List of Figures

1.1	Perception in the Robot-Environment exchange	4
1.2	Point Cloud Example	5
2.1	LiDAR Odometry Approaches	10
3.1	SDX-Compact	14
3.2	Example dataset trajectory	16
3.3	Data frame synchronisation	16
3.4	Motion compensation: before and after	18
3.5	KISS ICP Architecture	20
3.6	Trajectory deviation from GPS	20
3.7	Rotation and translation averaging experiment	24
3.8	Motion model challenges	26
3.9	Particularities of LiDAR output	27
3.10	Robust residual function	29
3.11	Filtering using entropy	31
3.12	Factor Graph Structure	33
3.13	Solution Architecture	34
4.1	Voxel size effect on map quality	39
4.2	Trajectory evaluation with and without GICP	40
4.3	Map evaluation with and without GICP	41
4.4	Odometry evaluation on custom trajectory	42
4.5	Position analysis on custom trajectory	42
4.6	Odometry evaluation on KITTI sequences	43
4.7	Trajectory corrections over noisy GPS	45

Acronyms

ATE Absolute Trajectory Error.

CDF Cumulative Distribution Function.

CNN Convolutional Neural Network.

DoF Degrees of Freedom.

EKF Extended Kalman Filter.

GICP Generalized ICP.

GLONASS Global Navigation Satellite System (Russian).

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

ICP Iterative Closest Point.

IMU Inertial Measurement Unit.

INS Inertial Navigation System.

KF Kalman Filter.

LiDAR Light Detection and Ranging.

LIO LiDAR-Inertial Odometry.

LOAM LiDAR Odometry and Mapping.

NDT Normal Distributions Transform.

PF Particle Filter.

PTP Precision Time Protocol.

RMSE Root Mean Squared Error.

RTE Relative Trajectory Error.

RTK Real-Time Kinematics.

SfM Structure from Motion.

SLAM Simultaneous Localization and Mapping.

SVD Singular Value Decomposition.

UDP Universal Datagram Protocol.

Glossary

Field Robotics The area of robotics that focuses on robots operating in unstructured outdoor environments for tasks like agriculture or exploration.

GNSS fix The desirable situation in which a connection to the satellite system is established, such that live localization information can be transmitted.

k-d tree A data structure for organizing entries in a k -dimensional space using a tree structure. When k is not very large, it provides logarithmic look-up time.

keyframe An image frame or 3D scan that is used for odometry estimation. Because modern sensors usually operate at higher frequencies than needed for most algorithms, it is a reasonable decision to skip frames based on a predefined pattern or condition, without sacrificing accuracy.

odometry The process of computing relative displacement of a robot using on-board sensors.

point cloud A set of point coordinates obtained as the output of a scanning sensor.

registration The process of aligning multiple point clouds in a single coordinate system such that matching features are as close as possible.

sensor fusion The process of combining information from multiple sensors, to augment the knowledge about the environment and increase the level of autonomy.

Structure from Motion (SfM) The task of reconstructing a 3D scene and a sequence of camera poses from a set of images.

surveying The work of examining and recording the area and features of a piece of land in order to construct a map, plan, or detailed description thereof.

voxel A cuboid-shaped region in 3D space; a 3D cell.

6

**DEEP REINFORCEMENT LEARNING FOR DRONE
OBSTACLE AVOIDANCE**
Thanh Loc Pham

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING



MASTER THESIS No.153

DEEP REINFORCEMENT LEARNING FOR DRONE OBSTACLE AVOIDANCE

Thanh Loc Pham

Erasmus Mundus Joint Master's Degree

Supervisors:

Dr. Goran Vasiljević

Assistant Professor

Benjamin van Manen

MSc, PhD Candidate

Zagreb, June, 2025

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA



DIPLOMSKI RAD br.153

**DUBOKO PODRŽANO UČENJE ZA
IZBJEGAVANJE PREPREKA POMOĆU
BESPILOTNIH LETJELICA**

Thanh Loc Pham

Erasmus Mundus Joint Master's Degree

Nadzornik:

Dr. Goran Vasiljević

Docent

Benjamin van Manen

Magistar znanosti, doktorand

Zagreb, lipanj, 2025.

I would like to express my sincere gratitude to the IFRoS program, the University of Girona, the University of Zagreb, and all partner institutions for granting me the opportunity to pursue this master's program. I appreciate not only the advanced robotics knowledge I gained, but also the way the curriculum was designed—it provided me with a broad, structured overview of the robotics field. This approach helped spark my interest, fueled my passion for robotics, and gave me a clear direction for my future career.

My deepest thanks go to my supervisors, Prof. Goran Vasiljević, and Ph.D. candidate Benjamin van Manen, for their guidance, straightforward feedback, and ongoing support throughout this project, as well as for their valuable advice regarding my future career path.

Special thanks to Dr. Imma Güell and Prof. Tamara Petrović for their roles in program coordination and student representation, and for their enthusiastic support with paperwork, housing, and helping me adjust to the academic and daily life during my first days in Spain (Year 1) and Croatia (Year 2). I truly appreciate this assistance, especially as someone studying abroad for the first time.

Finally, I am grateful to my colleagues, friends, family and girlfriend whose encouragement and honesty have helped me navigate the inevitable challenges of this journey.

Abstract

Unmanned Aerial Vehicles (UAVs) are increasingly deployed in autonomous missions across complex, cluttered environments where reliable obstacle avoidance is critical. Traditional navigation frameworks rely on modular pipelines—separating perception, mapping, planning, and control—which often suffer from error accumulation, high computational overhead, and poor reactivity in dynamic scenarios. To address these limitations, this thesis investigates an end-to-end deep reinforcement learning (DRL) framework for real-time UAV obstacle avoidance using onboard depth sensing.

We compare two state-of-the-art DRL algorithms, Proximal Policy Optimization (PPO) and Twin Delayed DDPG (TD3), in a continuous control setting, evaluating their training dynamics and performance in diverse simulated environments. Our initial experiments highlight key failure modes such as collisions with overhead obstacles and dead-end traps, caused by the policy's limited temporal awareness. To overcome these, we propose a neural architecture that incorporates both a pretrained ResNet8-based depth encoder and two temporal reasoning mechanisms: (1) an LSTM module for recurrent memory, and (2) a stacked buffer of recent depth observations. This temporal augmentation allows the agent to recover from occlusions and partial observability, significantly improving navigation robustness.

Trained in a curriculum-based Gym-PyBullet-Drones environment, our final memory-based policy achieves a 96% success rate across randomized 3D obstacle courses and outperforms EGO-Planner-v2 in both success rate and adaptability. The results demonstrate that DRL policies with temporal context can match or exceed the performance of traditional planning pipelines while offering greater generalization and simplicity in deployment.

Contents

Abstract	1
List of Figures	4
List of Tables	6
1 Introduction	7
1.1 Scope	8
1.2 Contributions	8
1.3 Outline	9
2 Related work	10
2.1 Traditional and Early Learning-Based UAV Obstacle Avoidance Approaches	10
2.2 Deep Reinforcement Learning for UAV Obstacle Avoidance	12
2.3 Recurrent Neural Networks for Partially Observable UAV Navigation	14
3 Background in Reinforcement Learning	17
3.1 Markov Decision Process (MDP)	18
3.1.1 The Agent-Environment Interface	18
3.1.2 Returns and Episodes	19
3.1.3 Policies and Value Functions	20
3.1.4 Optimal Policies and Optimal Value Functions	20
3.2 Tabular Reinforcement Learning and Limitations	21
3.3 Taxonomy of RL Algorithms	22
3.4 Deep Reinforcement Learning	24
3.4.1 Three Challenges	24
3.4.2 Trust Region Optimization, TRPO and PPO	25

3.4.3	Deterministic Policy Gradient, DDPG and TD3	25
4	Methodology	27
4.1	Problem Formulation	27
4.2	Reinforcement Learning	29
4.2.1	Reward Function Design	30
4.2.2	Neural Network Architecture	32
5	Experiments and Results	36
5.1	Experiments Setup	36
5.1.1	Drone Configuration	37
5.1.2	Start and Goal Placement	38
5.1.3	Obstacle Design	38
5.1.4	Low-Level Controller	39
5.1.5	Curriculum Learning	40
5.1.6	Noise Injection for Realism and Robustness	42
5.1.7	Reinforcement Learning Parameters	43
5.2	Results	44
5.2.1	PPO vs. TD3 with Basic Network Architecture	44
5.2.2	Effect of Temporal Features: Memory-Based Architecture	46
5.2.3	Comparison with Conventional Planning	49
6	Conclusion and Future Work	51
	References	54

List of Figures

3.1	Simple block-scheme representation of MDP (image taken from [45]). . .	18
4.1	Drone coordination and State Observation	28
4.2	Reward Shaping	30
4.3	Architecture of the Basic Network	32
4.4	Architecture of the Memory-based Network	34
4.5	Dead-end Case Example with depth images captured at the position 1, 2, 3, 4, 5, 6 from the left to the right.	35
5.1	Map setup: Red point at the center of the map: drone starting point; Green point: goal which randomly sampled around the map	38
5.2	Static obstacle types: blocks, walls, spheres, and a U-shaped dead-end. . .	39
5.3	5 levels of map complexity. (a) Map level 1: empty map, red point is the drone starting point, green points are the goals; (b) Map level 2: add verti- cal and horizontal cylinders; (c) Map level 3: add vertical cubes; (d) Map level 4: add walls and corners; (e) Map level 5: add dead-end obstacles (U shape).	41
5.4	PPO vs. TD3 with Basic Network Architecture Result Comparisons. . . .	45
5.5	Drone path with PPO on the train and test map. Green path: Path suc- cessfully reaches the goal; Red path: Path collides	46
5.6	Overhead and underfoot obstacles crash case.	46
5.7	Dead-End Traps. The green path is the drone path when it goes into a dead-end obstacle with U-shape and it gets collision at the end of the path. .	47
5.8	Training progress of the memory-based architecture.	47
5.9	Drone path in complex environment (train map level 5). Green path: Path successfully reaches the goal; Red path: Path collides	48

5.10 The drone path with the underfoot obstacle.	48

List of Tables

4.1	Pretrained NTNU ResNet8 Feature Extractor Architecture	33
5.1	Drone Configuration	37
5.2	Environment Parameters	38
5.3	Reward Function Parameters and Values	42
5.4	Noise Parameters	43
5.5	Training Parameters	43
5.6	Basic architecture performance in training and testing map.	45
5.7	Performance Comparison in Training and Testing Environments	49

1 Introduction

Unmanned Aerial Vehicles (UAVs) have gained increasing attention for their potential in autonomous navigation across diverse and dynamic environments. One of the key challenges in real-world deployment is reliable obstacle avoidance, particularly in unknown and cluttered spaces. Traditional approaches to autonomous UAV navigation often follow a cascaded design, involving separate modules for perception, mapping, path planning, and control. Visual navigation frameworks in this category typically reconstruct 3D maps from depth images and utilize them for subsequent planning and control tasks. Examples such as Fast-Planner and EGO-Planner-v2 demonstrate the effectiveness of this modular pipeline, employing depth-based grid mapping, trajectory optimization, and visual-inertial odometry (VIO) to achieve precise trajectory tracking and control.

However, despite their interpretability and structured design, these traditional methods face several limitations. The modular separation can result in error accumulation, as downstream modules depend heavily on the accuracy of upstream outputs. Additionally, constructing and maintaining detailed maps consumes significant computational resources, often leading to reduced responsiveness in dynamic or highly cluttered environments—precisely the conditions where agility and reactivity are most critical for UAV operation.

To address these challenges, recent research has explored end-to-end learning approaches, particularly using deep learning. End-to-end frameworks based on supervised learning or reinforcement learning can directly learn control policies from visual inputs, bypassing the need for intermediate mapping or hand-crafted planning heuristics. While such models may sacrifice some interpretability, they offer several advantages: reduced computational overhead, improved real-time performance, and better generalization across diverse environments. Among these, Deep Reinforcement Learning (DRL)

stands out for its ability to learn complex behaviors from interaction with the environment, guided by reward functions that capture mission objectives such as safety, efficiency, and goal completion.

1.1 Scope

In this thesis, we focus on a DRL-based framework for UAV obstacle avoidance which is inspired by the work of Mihir et al. [1] and Hang Yu et al. [2]. This representation is explicitly trained to retain both current and past depth observations, enabling the policy to infer a compact internal model of the environment.

1.2 Contributions

1. **Integrated perception and control pipeline:** We propose a unified DRL-based policy that directly maps raw depth images and relative goal information to high-level UAV control commands, eliminating the need for explicit mapping, path planning, or trajectory optimization modules.
2. **Temporal visual encoding:** To address partial observability, we introduce a neural architecture that fuses latent features from a sequence of past depth images. This enables the policy to maintain spatial awareness over time, particularly critical for avoiding dead-end obstacles and handling occluded threats like overhead collisions.
3. **Recurrent memory integration:** We augment the policy with an Long short-term memory (LSTM) module, allowing it to learn long-term dependencies and internal representations of the environment, further improving navigation robustness in cluttered or dynamic scenarios.
4. **Comprehensive evaluation and comparison:** We conduct extensive experiments comparing two DRL algorithms—Proximal Policy Optimization (PPO) and Twin Delayed DDPG (TD3)—under varying architecture configurations, and benchmark our final model against EGO-Planner-v2, a state-of-the-art conventional trajectory planning method.

5. **Simulation environment design:** We build a customizable and curriculum-compatible training environment using Gym-PyBullet-Drones, including procedurally generated maps with adjustable obstacle types and complexity levels, to promote generalization and policy transferability.

Overall, this thesis contributes a scalable, flexible, and real-time obstacle avoidance solution that combines the strengths of deep reinforcement learning with a lightweight memory mechanism, bridging the gap between traditional map-based planning and modern end-to-end learning approaches.

1.3 Outline

This thesis is organized into six chapters. Chapter 1 introduces the motivation for vision-based UAV obstacle avoidance and discusses the limitations of traditional modular navigation systems. It also defines the scope and key contributions of this work. Chapter 2 reviews related literature, covering both classical and learning-based navigation approaches, with a focus on deep reinforcement learning methods and the integration of temporal features. Chapter 3 provides foundational background in reinforcement learning, starting from basic tabular methods and progressing to advanced algorithms such as PPO and TD3. Chapter 4 describes the proposed methodology, including the simulation environment, neural network architecture, temporal feature integration, reward design, and training setup. Chapter 5 presents experimental results, comparing different network configurations, analyzing failure cases, and benchmarking the final policy against EGO-Planner-v2. Finally, Chapter 6 summarizes the main findings, discusses limitations, and outlines directions for future research.

2 Related work

This chapter reviews prior research relevant to drone obstacle avoidance and introduces deep reinforcement learning approaches for this task, the remaining problems in the research direction and some solutions for them.

2.1 Traditional and Early Learning-Based UAV Obstacle Avoidance Approaches

Unmanned Aerial Vehicle (UAV) obstacle avoidance has long been a focus of robotics research, with early solutions predominantly relying on traditional (non-learning) methods. These classical approaches can be broadly categorized into graph-based path planners, reactive local methods, and sensor-based control schemes. Graph-based planners (e.g., grid search algorithms like Dijkstra's [3], A* algorithm in [4] or sampling-based planners like Rapidly-Exploring Random Trees, RRT in [5]) generate collision-free routes on a map. Such methods are effective in static environments; for example, Dijkstra's algorithm was adapted for UAVs to plan the shortest paths around known obstacles. However, purely global planners struggle with dynamic obstacles or unknown environments, as they assume a pre-mapped or static world. Although there have been many studies aimed at improving the performance of these methods to solve problems with dynamic environments such as D* algorithm was proposed by Stentz at Carnegie Mellon Robotics Center in 1994 and is mainly used for robotic path finding [6]; [7] proposed a UAV path planning algorithm ADRRT*-connect for dynamic obstacles in 3D environments that requires only 3.5% of new nodes to repair the paths in re-planning, saving the path planning cost, the balance between performance and accuracy is still a big problem with these algorithms.

In contrast, reactive methods make instantaneous decisions based on sensor read-

ings. Classic examples include the Bug algorithm [8], which steers the UAV around encountered obstacles, and artificial potential fields, which treat obstacles as repulsive forces to be avoided. These techniques enable real-time avoidance without a global map, though they may get trapped in local minima or dead-ends if not carefully designed. Variants like vector field histograms [9] and dynamic window [10] approaches also fall in this category, providing fast obstacle clearance by following the gradient of a constructed potential or velocity field.

Lastly, sensor-based control schemes integrate multiple sensors and control laws (often in a hierarchical loop) to improve reliability. For instance, sense-and-avoid systems fuse data from vision, LiDAR/radar, and ultrasonics to detect obstacles in all directions, feeding into a low-level controller or model predictive controller (MPC) that adjusts the UAV's trajectory to avert collisions. Similarly, an MPC method with an occupancy grid filter allowed a drone to predict obstacle motion beyond its immediate field of view, enhancing safety in cluttered spaces [11]. These traditional techniques established a foundation for UAV autonomy, though they often require careful tuning and still face limitations under partial observability and highly dynamic conditions.

Early attempts to improve UAV obstacle avoidance began incorporating machine learning well before the deep learning era. One line of work explored supervised learning and imitation: for example, DroNet by Loquercio et al. is a convolutional neural network trained on urban driving images to output steering commands for a drone, effectively learning to “fly by driving” [12]. This network enables a drone to cruise along city streets while recognizing obstacles and avoiding collisions by mimicking the reactive behaviors of cars and bicycles and similar vision-based policies demonstrated that neural networks could map camera inputs to safe control outputs, reducing reliance on manual feature design. Another stream used classic reinforcement learning (RL) with small state representations. Prior to deep RL, researchers investigated tabular or simple neural-network RL for UAVs, but these were limited by state dimensionality and sample inefficiency. Some works also applied evolutionary algorithms or fuzzy logic controllers to tune avoidance behaviors automatically. While these learning-based approaches showed promise, their success often hinged on instrumented environments or extensive offline training. For instance, one study attempted to train a drone's policy on a large dataset of indoor

images (varied lighting, layouts) using a deep Q-network (DQN) to predict motions [13]. In practice, this offline-trained network struggled to generalize – despite thousands of hallway images, it failed to achieve reliable navigation, highlighting that static image datasets cannot capture the interactive, sequential nature of flight. The authors noted this approach “did not mimic human learning behavior” and lacked adaptiveness. These early limitations motivated a shift toward more powerful deep reinforcement learning methods that learn by trial-and-error in simulation or real life, as discussed next.

2.2 Deep Reinforcement Learning for UAV Obstacle Avoidance

Building on breakthroughs in deep reinforcement learning, recent research has extensively applied DRL algorithms to autonomous drone navigation and obstacle avoidance. In the DRL paradigm, UAV control is formulated as a sequential decision-making problem: the agent (drone) observes its environment state, takes an action (e.g. adjust velocity or heading), and receives a reward signal indicating progress or collisions, with the goal of maximizing cumulative reward. This framework has enabled UAVs to learn avoidance strategies that would be difficult to hand-engineer. Early successes employed value-based methods like Deep Q-Learning (DQL), which uses neural networks to approximate the Q-value of state–action pairs [14]. Extensions of DQN for UAVs introduced techniques like prioritized experience replay and double Q-learning to stabilize training under stochastic conditions [15]. Researchers have also integrated customized heuristics into RL; one framework augmented Q-learning with a shortest-distance heuristic, encouraging the agent to favor locally efficient detours during avoidance [16]. Beyond discrete action Q-learning, continuous control algorithms have gained popularity for smooth UAV motion. Deep Deterministic Policy Gradient (DDPG), an actor-critic method, was utilized in a self-trained UAV that learned to avoid both moving and static obstacles in a cluttered 3D urban simulation [17]. Such actor-critic methods output continuous control commands (e.g. thrust or turn rate), suiting the UAV’s real-valued action space better than discretized steering commands. Among deep RL algorithms, two families have emerged as particularly effective for UAV navigation: on-policy methods like Proximal Policy Optimization (PPO) and off-policy actor-critics like Twin Delayed Deep De-

terministic Policy Gradient (TD3). PPO is an on-policy algorithm that optimizes a policy by iterative interactions with the environment, using a clipped surrogate objective to ensure stable updates. Its appeal lies in its training stability and simplicity – PPO has been shown to learn robust UAV navigation policies without requiring complex hyperparameter tuning for Q-value estimation. On the other hand, TD3 is an advanced off-policy algorithm built upon DDPG, addressing the overestimation bias in value estimates via twin Q-networks, delayed policy updates, and target noise. TD3’s ability to handle continuous actions with improved stability makes it well-suited for drone control tasks that demand smooth and precise maneuvers.

Several studies have explicitly compared PPO and TD3 in UAV scenarios. In a recent experimental study in a realistic simulator (Unreal Engine + AirSim), TD3 outperformed PPO on various metrics for single-drone obstacle avoidance [18]. This suggests that off-policy learning with value function correction can yield more efficient policies for navigation. However, the performance ranking of algorithms can depend on task specifics and reward design. In general, actor-critic methods (DDPG, TD3) excel in continuous control of UAVs, while on-policy methods (PPO, A3C) provide robustness and ease of deployment; the choice often depends on whether sample efficiency (favoring off-policy) or training stability (favoring on-policy) is the priority. Beyond algorithm selection, researchers have incorporated various enhancements to boost RL performance for drones. For example, combining representation learning with RL has shown benefits: one study used a variational auto-encoder (VAE) to preprocess high-dimensional images into a compact latent state, which converged faster and reached higher rewards than a baseline that fed raw images directly [1] [2]. Reward shaping and safety constraints are also employed to guide learning – e.g., penalties for proximity to obstacles or time-out conditions for getting stuck in corners. Multi-agent extensions of deep RL have been explored for drone swarms or crowded airspace, though the thesis at hand focuses on a single agent. While multi-agent reinforcement learning opens questions of coordination and competition, single-UAV scenarios allow us to isolate and compare algorithmic performance (like PPO vs TD3) more directly. In summary, deep RL has become a cornerstone of modern UAV obstacle avoidance research, enabling end-to-end learning of navigation policies. The next sections discuss two key ingredients for improving deep RL in this context: recurrence for handling partial observability, and enhanced perception

via depth and temporal sensing.

2.3 Recurrent Neural Networks for Partially Observable UAV Navigation

A critical challenge in drone navigation is partial observability. Unlike in a fully-known map, a UAV typically only observes its immediate surroundings through onboard sensors with limited range or field of view. The environment's true state (e.g. the layout of all obstacles) is only partially observable at any given time, making the problem a Partially Observable Markov Decision Process (POMDP). In such scenarios, an optimal decision at the current step may require memory of past observations — for example, a drone rounding a corner needs to remember whether there was a dead-end on one side, or how narrow the passage behind it was [13]. Without memory, the UAV could blindly fly into a trap or fail to backtrack from a cul-de-sac because it “forgets” what it saw a few moments ago. Human pilots naturally integrate temporal context and recall recently seen obstacles, which has inspired researchers to equip autonomous drones with similar capabilities.

Recurrent neural networks (RNNs) have become a popular solution to provide the RL agent with an internal memory of past states. By incorporating an RNN (such as an LSTM or GRU) into the policy or value network, the UAV can maintain a hidden state that aggregates information over time. This memory enables more informed decisions in ambiguous situations, effectively augmenting the observation with a short-term history. Singla et al. (2019) [13] demonstrated a compelling example of this in a work titled “Memory-based Deep Reinforcement Learning for Obstacle Avoidance in UAVs with Limited Environment Knowledge.” They propose a deep RL approach that explicitly addresses partial observability by letting the UAV retain relevant information about the environment structure to make better future navigation decisions. This recurrent policy significantly improved the drone's performance in unknown indoor environments, allowing it to cover longer distances without collision compared to a memoryless (feed-forward) baseline. The idea is that as the UAV flies, the LSTM accumulates knowledge of the layout – for instance, remembering that a certain corridor was blocked – and the attention module helps recall that specific memory when the UAV revisits that area or

needs to decide on an alternative route.

The importance of memory for UAV navigation is echoed by other studies. Hausknecht and Stone’s Deep Recurrent Q-Network (DRQN) approach (originally applied in video games) laid the groundwork for integrating recurrence into Q-learning, and similar principles have been adapted for drones. For example, a DRQN with a temporal attention module was applied to UAV obstacle avoidance in unseen environments, showing that the agent could effectively handle moving obstacles and occluded hazards by leveraging its internal state [19]. The UAV agent, in essence, learns a policy over belief states (implicit estimates of true state given observation history) rather than raw instantaneous observations. In more recent developments, researchers are even exploring transformer-based sequence models as an alternative to RNNs for capturing long-range dependencies in observation sequences. For instance, a 2024 study used a dual-transformer encoder to process sequences of UAV observations in a target tracking task, achieving superior trajectory optimization and obstacle avoidance performance compared to LSTM-based critics [20]. While transformers have yet to be widely implemented in drone collision avoidance, they represent a growing interest in leveraging advanced sequence modeling for partial observability.

Another noteworthy approach to memory in RL is the use of reservoir computing or echo state networks (ESN) within the agent. Unlike standard trained RNNs, ESNs provide a fixed recurrent layer with dynamic memory properties. One study formulated the UAV navigation problem as a dynamic game and employed an echo state network-based deep RL algorithm, which guaranteed convergence to a Nash equilibrium in a multi-UAV scenario. Each drone’s policy network included ESN cells to capture the temporal context of interactions (like interference with ground stations and other UAVs), enabling adaptive and generalized trajectory planning without requiring the drone to directly observe all factors at once. This highlights that memory mechanisms in UAV RL can take various forms—from learned LSTMs to fixed dynamical systems—all aiming to surmount the limitations of partial observability.

In summary, incorporating recurrence or memory is crucial when the UAV must navigate complex, partially known environments. By remembering past observations, the agent can avoid “blind” decisions. This is especially vital for avoiding dead-end scenar-

ios: if a drone enters a dead-end alley, a memory-enabled policy can recall how it got in and plan a retreat, whereas a reactive memoryless policy might oscillate or crash. Empirical evidence across studies indicates that memory-based RL agents significantly reduce collisions in such scenarios [13]. As UAV obstacle avoidance tasks often involve exploration of unseen areas, the use of RNNs or other sequence models has become a standard component of advanced solutions.

3 Background in Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm in which an agent learns to map situations to actions in order to maximize a cumulative reward. Unlike supervised learning, where correct actions are provided by a labeled dataset, the RL agent must learn through trial and error, discovering which actions yield the most reward through interaction with the environment. A distinguishing feature of RL is the notion of *delayed rewards*, where actions can influence not only immediate outcomes but also future states and rewards.

The RL framework is typically formalized using *Markov Decision Processes* (MDPs), which capture the agent-environment interaction in terms of states, actions, transitions, and rewards. The agent senses the current state, takes actions that influence the environment, and aims to reach goals defined via a reward function. This structure allows RL to model sequential decision-making problems in a principled way.

RL differs from *supervised learning*, which generalizes from labeled examples, and from *unsupervised learning*, which discovers structure in unlabeled data. RL constitutes a third paradigm, focused on maximizing reward through interaction rather than fitting labels or uncovering hidden patterns. While representation learning may support RL, it is not sufficient for solving RL problems on its own.

A key challenge in RL is the *exploration-exploitation dilemma*: the agent must explore new actions to discover better strategies while exploiting known actions to gain reward. Balancing this trade-off is fundamental to effective learning and does not arise in conventional supervised or unsupervised settings.

Unlike many machine learning approaches that address isolated subproblems (e.g., classification or planning), RL addresses the entire loop of sensing, acting, and learning

in uncertain environments. This makes RL particularly relevant for real-world decision-making tasks.

Finally, RL has strong interdisciplinary links. It overlaps with control theory through its optimization focus, leverages statistical methods for estimation, and draws inspiration from neuroscience and psychology. Several RL algorithms have roots in models of animal and human learning, and RL continues to inform both theoretical and empirical research across disciplines.

3.1 Markov Decision Process (MDP)

Having introduced the general concept and formalism of Markov Decision Processes (MDPs), we now turn to the concrete components that define the agent's interaction with its environment. The following subsections break down each aspect of this interface, laying the groundwork for how reinforcement learning problems are modeled and solved in practice.

3.1.1 The Agent-Environment Interface

Reinforcement Learning problems are commonly formalized using Markov Decision Processes (MDPs). The learner and decision maker is called the *agent*. The thing it interacts with, comprising everything outside the agent, is called the *environment*. These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent seeks to maximize over time through its choice of actions.

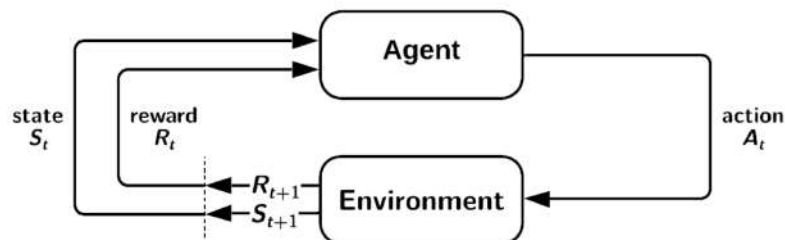


Figure 3.1: Simple block-scheme representation of MDP (image taken from [45]).

An MDP is a discrete-time stochastic model defined by the tuple (S, A, P, R) , where:

- $S_t \in \mathcal{S}$: Set of environment states that represent the configuration of the system at a certain time instant. An important property of the state variables is that they allow to fully know the system configuration without requiring any information about the system past history. It is *Markov property*.
- $A_t \in \mathcal{A}(s)$: Set of actions available to the agent from any states. The set is indicated with $\mathcal{A}(s)$, which represents the set of actions that can be taken from state s . Both $s \in \mathcal{S}$ and $a \in \mathcal{A}$ can be continuous or discrete variables, depending on the definition of the problem.
- P : Transition probability function defines the *dynamics* of the MDP $P(s'|s, a) = P(S_{t+1} = s' | S_t = s, A_t = a)$. The probability transition function implies that, in general, the MDP is a stochastic process, since there is no way to determine the state at time $t + 1$ other than through a probability distribution. However, in some situations (like the one that will be analysed later in the algorithm) the environment is fully deterministic and the state at time $t + 1$ can be derived from S and A at time t , so $P(s'|s, a) = 1$ only for the state s' and 0 for all the others.
- $R_t \in \mathcal{R}$: Reward function $R_a(s, s')$ indicating the reward for transitioning from state s to s' via action a . For any combination of s, a, s' the function returns $R = R(s'|s, a)$

3.1.2 Returns and Episodes

If the sequence of rewards received after time step t is denoted $R_{t+1}, R_{t+2}, R_{t+3}, \dots$, then what precise aspect of this sequence do we wish to maximize? In general, we seek to maximize the expected return, where the return, denoted G_t , is defined as some specific function of the reward sequence. Regularly, the discounted cumulative reward is added in order to take into account long-term strategy by the agent. The discounted cumulative reward function (also called expected discounted return) is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R_t + \gamma G_{t+1} = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots \quad (3.1)$$

where R_i is the reward obtained at time step $i - 1$ selecting an action through the policy. The discount factor $\gamma < 1$ gradually reduces the importance of future rewards in the sum. Depending on the type of policy that is used, the maximization of the expected

return is obtained following different strategies. In the next sections, the RL training process through which this goal is achieved will be introduced.

3.1.3 Policies and Value Functions

Almost all reinforcement learning algorithms involve estimating *value functions*—functions of states (or of state–action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). Meanwhile, a policy is a mapping from states to probabilities of selecting each possible action. If the agent is following policy π at time t , then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

The value function of a state s under a policy π , denoted $V_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, we can define V_π formally by

$$V_\pi(s) \doteq \mathbb{E}_\pi [G_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \text{ for all } s \in \mathcal{S} \quad (3.2)$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of a random variable given that the agent follows policy π , and t is any time step. Note that the value of the terminal state, if any, is always zero. We call the function V_π the state-value function for policy π .

Similarly, we define the value of taking action a in state s under a policy π , denoted $Q_\pi(s, a)$, as the expected return starting from s , taking the action a , and thereafter following policy π :

$$Q_\pi(s, a) \doteq \mathbb{E}_\pi [G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \quad (3.3)$$

We call Q_π the action-value function for policy π .

3.1.4 Optimal Policies and Optimal Value Functions

Solving a reinforcement learning task means, roughly, finding a policy that achieves a lot of reward over the long run. For finite MDPs, we can precisely define an optimal policy in the following way. Value functions define a partial ordering over policies. A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geq \pi'$ if and only if $V_\pi(s) \geq V_{\pi'}(s)$

for all $s \in \mathcal{S}$. There is always at least one policy that is better than or equal to all other policies. This is an optimal policy. Although there may be more than one, we denote all the optimal policies by π_* . They share the same state-value function, called the optimal state-value function, denoted v_* , and defined as

$$V_*(s) \doteq \max_{\pi} V_{\pi}(s) \quad (3.4)$$

for all $s \in \mathcal{S}$.

Optimal policies also share the same optimal action-value function, denoted q_* , and defined as

$$Q_*(s, a) \doteq \max_{\pi} Q_{\pi}(s, a), \quad (3.5)$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$.

3.2 Tabular Reinforcement Learning and Limitations

Tabular reinforcement learning methods are foundational approaches that assume a finite and discrete state-action space. These methods maintain explicit tables to store value functions or policies, making them suitable for small-scale problems. The most well-known tabular methods include *Dynamic Programming (DP)*, *Monte Carlo (MC)* methods, and *Temporal-Difference (TD)* learning. All three follow the goal of estimating value functions and optimizing policies, but they differ significantly in how they acquire knowledge and update values.

While DP's reliance on a perfect model and high computational demands limit its practical applicability to small domains. Monte Carlo methods avoid the need for a known model by learning directly from sampled experience. They estimate value functions by averaging observed returns following visits to states (or state-action pairs). However, because updates only occur at the end of episodes, MC methods can be inefficient for long or continuous tasks. TD methods are more sample-efficient than MC and suitable for online, incremental learning. In control settings, TD learning gives rise to *on-policy* methods like **SARSA**, which learns the value of the policy it follows, and *off-policy* methods like **Q-learning**, which learns the value of the optimal policy independently of the behavior policy.

Tabular methods do not scale well to high-dimensional problems due to two main issues:

- **Scalability:** As the number of states and actions increases, storing and updating exact values becomes computationally intractable.
- **Lack of Generalization:** Learning in one state does not transfer to similar states, making learning inefficient in continuous or structured environments.

To overcome these, function approximators—such as linear models or neural networks—are used to estimate value functions or policies. While linear models are suitable for these problems simple and converge reliably but lack expressiveness. The nonlinear approximators like deep neural networks can model complex patterns and scale to unstructured inputs. They are more powerful but may suffer from instability and overfitting. Depending on the input type, different architectures are used:

- *MLPs* for structured low-dimensional input.
- *CNNs* for visual or spatial data.
- *RNNs* for sequential or partially observable tasks.

The success of deep Q-networks (DQNs) in visual tasks like Atari games illustrates the power of function approximation in reinforcement learning.

3.3 Taxonomy of RL Algorithms

The development of reinforcement learning can be divided into three main approaches: value-based, policy-based, and actor-critic methods. An overview of these three approaches can be presented as follows.

Value-based methods find the policy in a two-step process. First, they find the best action-value of a state, for which then the accompanying actions are found (by means of $\arg \max$). This works in environments with discrete actions, where the highest-valued action is clearly separate from the next-best action. Examples of continuous action spaces are robot arms that can move over arbitrary angles, or poker bets that can be of any

monetary value. In these action spaces, value-based methods become unstable and *arg max* is not appropriate.

Another approach works better: policy-based methods. Policy-based methods do not use a separate value function but find the policy directly. They start with a policy function, which they then improve, episode by episode, with policy gradient methods. Policy-based methods are applicable to more domains than value-based methods. They work well with deep neural networks and gradient learning; they are some of the most popular methods of deep reinforcement learning. One of them is REINFORCE.

The policy-based method introduced many advantages. First, the parameterization is at the core of policy-based methods, making them a good match for deep learning. For value-based methods deep learning had to be retrofitted, giving rise to complications. Second, policy-based methods can easily find stochastic policies; value-based methods find deterministic policies. Due to their stochastic nature, policy-based methods naturally explore, without the need for methods such as ϵ -greedy, or more involved methods, that may require tuning to work well. Third, policy-based methods are effective in large or continuous action spaces. Small changes in θ lead to small changes in π , and to small changes in state distributions (they are smooth).

However, in REINFORCE, the evaluation of policies based on full trajectories has low sample efficiency and high variance. As a consequence, policy improvement happens infrequently, leading to slow convergence compared to value-based methods. Moreover, this approach often finds a local optimum since convergence to the global optimum takes too long. One solution is the Actor-Critic approach which combines value-based elements with the policy-based method was introduced. The actor stands for the action, or policy-based, approach; the critic stands for the value-based approach. The variance of policy methods can originate from two sources: (1) high variance in the cumulative reward estimate, and (2) high variance in the gradient estimate. For both problems a solution has been developed: bootstrapping for better reward estimates, and baseline subtraction to lower the variance of gradient estimates.

3.4 Deep Reinforcement Learning

Deep Reinforcement Learning (Deep RL) is a paradigm that integrates reinforcement learning (RL) algorithms with deep neural networks, enabling agents to learn complex policies directly from high-dimensional sensory inputs such as images, audio, or raw sensor data. This integration has propelled RL beyond its traditional scope—where tabular or linear methods dominated—to tackle challenging problems in robotics, game playing, autonomous driving, and more.

3.4.1 Three Challenges

Let us have a closer look at the challenges that deep reinforcement learning faces. There are three problems with our naive deep Q-learner.

1. Insufficient State Space Coverage Convergence of value-based methods like Q-learning relies on the assumption that all state-action pairs are visited sufficiently often. However, in large or continuous environments, exhaustive coverage is infeasible.

2. Correlated Samples and Non-Stationarity Unlike supervised learning, where training samples are i.i.d., RL collects data sequentially through agent-environment interaction. This causes strong correlations between consecutive states and introduces bias. Additionally, as the policy changes during training, the distribution of visited states shifts, making the learning target itself a moving objective.

3. Instability Due to Bootstrapping Bootstrapping methods like Temporal-Difference (TD) learning update estimates based on other learned estimates. When combined with deep networks, this recursive dependency can lead to divergence if value estimates are inaccurate.

Together, these challenges—limited exploration, data correlation, and bootstrapped target instability—necessitate careful algorithm design. Techniques such as experience replay, target networks, and constrained updates (e.g., PPO, TRPO) have been introduced to mitigate these issues.

3.4.2 Trust Region Optimization, TRPO and PPO

To address these challenges, some solutions were introduced, including Temporal Difference Bootstrapping and Baseline Subtraction with Advantage Function [3.3]. Another important approach to further reduce the variance of policy methods is the trust region approach. Trust region policy optimization (TRPO) [21] aims to further reduce the high variability in the policy parameters, by using a special loss function with an additional constraint on the optimization problem.

A naive approach to speed up an algorithm is to try to increase the step size of hyperparameters, such as the learning rate, and the policy parameters. This approach will fail to uncover solutions that are hidden in finer grained trajectories, and the optimization will converge to local optima. For this reason the step size should not be too large. A less naive approach is to use an adaptive step size that depends on the output of the optimization progress. Trust Region Policy Optimization (TRPO) addresses this by enforcing a constraint that new policies must remain close to the old ones. Specifically, TRPO maximizes a surrogate objective subject to a constraint on the Kullback-Leibler (KL) divergence in order to ensure monotonic improvement in policy performance and prevents destructive updates. However, TRPO relies on second-order optimization (via conjugate gradient), making it complex to implement and computationally heavy.

To address these limitations, Proximal Policy Optimization (PPO) [22] was introduced as a simpler yet effective alternative. PPO replaces the hard KL constraint with a clipped objective, which prevents excessively large policy updates while avoiding second-order derivatives.

Both algorithms are on-policy and widely used in continuous control benchmarks. PPO in particular has become a standard baseline in modern policy optimization due to its robustness and simplicity.

3.4.3 Deterministic Policy Gradient, DDPG and TD3

Actor critic approaches improve the policy-based approach with various value-based ideas, and with good results. Another method to join policy and value approaches is to use a learned value function as a differentiable target to optimize the policy against-we

let the policy follow the value function. An example is the deterministic policy gradient [23].

Deep Deterministic Policy Gradient (DDPG) [24] extends DPG with deep neural networks, experience replay, and target networks—techniques inspired by DQN. The actor and critic networks are trained using samples from a replay buffer, and target networks are updated slowly to improve stability.

Despite its success in continuous control tasks, DDPG suffers from overestimation bias and unstable training.

Twin Delayed DDPG (TD3) [25] improves DDPG through three key strategies:

- Double Q-learning: Maintains two critic networks and uses the minimum value to reduce overestimation.
- Delayed policy updates: Updates the actor less frequently than the critics to stabilize learning.
- Target policy smoothing: Adds noise to the target action to make the value function less sensitive to overfitting.

DDPG and TD3 are *off-policy* actor-critic algorithms and are widely used for high-dimensional robotic control. TD3, in particular, has become a strong baseline for stable and efficient learning in continuous action spaces.

4 Methodology

The last section introduced a background of reinforcement learning, how reinforcement learning problems are commonly formalized using Markov Decision Processes (MDPs) and the solutions to address those problems. In this section, we detail the drone navigation problem based on reinforcement learning, including problem formulation, the reward function design, and a basic neural network architecture for the drone navigation task and proposed memory-based architecture that can deal with more complex problems.

4.1 Problem Formulation

We model the obstacle avoidance task of an unmanned aerial vehicle (UAV) as a sequential decision-making problem under uncertainty, using the formalism of a Partially Observable Markov Decision Process (POMDP) (a tuple $\langle S, A, O, P, R, \gamma \rangle$) as mentioned in the above section. This framework captures the fact that the UAV has limited perceptual access to the environment and must make decisions based on incomplete observations.

1. **State (S):** Represents the full underlying environment configuration, including UAV position, velocity, orientation, nearby obstacles, and goal location. These states are not fully accessible to the agent due to sensor limitations.
2. **Observation (O):** The observation is a partial view of the state derived from on-board sensors. In our case, this includes a temporal stack of depth images captured from a forward-facing camera, along with a low-dimensional vector representing the relative displacement and orientation to the goal. The UAV must infer state information from these observations.

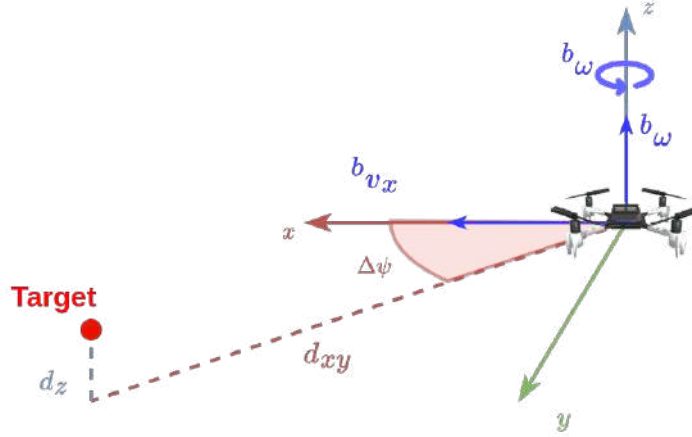


Figure 4.1: Drone coordination and State Observation

- A state vector describing the goal-relative position and orientation:

$$O_{\text{state}} = \begin{bmatrix} d_x \\ d_z \\ \Delta\psi \end{bmatrix} \quad (4.1)$$

where d_x and d_z denote the horizontal and vertical distance to the target, and $\Delta\psi$ is the yaw angle difference between the UAV's heading and the line-of-sight direction to the target.

- A visual input vector in the form of a stacked buffer of recent depth images, denoted as $O_{\text{image}} \in \mathbb{R}^{T \times H \times W}$, where T is the number of recent frames (temporal window), and (H, W) are the image dimensions. These images simulate a forward-facing depth camera and allow the agent to “see” obstacles and free space in front of the drone.

3. **Action (A):** The action is a continuous 3-dimensional control command:

$$A = \begin{bmatrix} b_{v_x} \\ b_{v_z} \\ b_{\omega} \end{bmatrix} \quad (4.2)$$

indicating the desired forward horizontal velocity, vertical velocity, and yaw rate in

the body frame. These actions are passed to a low-level controller that translates them into motor RPM commands.

4. **Transition Probability (P):** The transition dynamics are determined by the UAV's motion model and the physics simulated by PyBullet. This includes aerodynamic effects, inertia, external perturbations, and interactions with obstacles (e.g., collisions). The simulator models these effects deterministically but from the agent's perspective, they appear stochastic due to observation noise and partial observability.
5. **Reward (R):** The reward function is shaped to encourage goal-reaching behavior while penalizing collisions and inefficient trajectories. Specifically, the agent receives a high positive reward for reaching the goal, a large penalty for collisions, and intermediate shaping rewards based on distance reduction to the goal.
6. **Discount Factor (γ):** The discount factor balances short-term and long-term performance. A value close to 1 (e.g., $\gamma = 0.99$) encourages the agent to plan ahead and avoid short-sighted actions that may lead to collisions or dead ends.

In each episode, the UAV starts at a fixed location and must reach a goal placed at a constant distance away, navigating through randomly placed obstacles. Due to partial observability and delayed reward feedback, the agent must learn a policy $\pi(a|o)$ that maps its current observation to a control action, aiming to maximize the expected discounted return over time:

$$\mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

This formulation reflects the challenges of real-world navigation: the UAV must make decisions under uncertainty, reason over time using visual input and proprioceptive state, and avoid unsafe actions despite not having full access to the global environment state.

4.2 Reinforcement Learning

In this section, we present the core of the UAV autonomous navigation framework: the deep reinforcement learning policy network. We begin by introducing the reward function design tailored to this study, which plays a critical role in guiding the UAV's behavior.

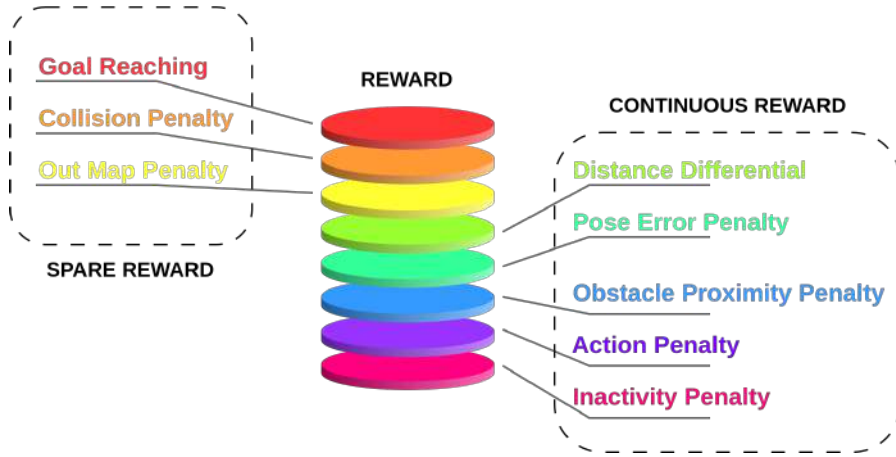


Figure 4.2: Reward Shaping

Next, we detail the integration of deep reinforcement learning with recurrent neural networks and image buffer, explaining how buffer information is utilized during training to enhance model performance, along with the policy update mechanism. Finally, we describe the specific network architecture designed to effectively implement the proposed framework and handle the challenges of partially observable environments.

4.2.1 Reward Function Design

In this work, we design a reward function that integrates both sparse and continuous components to guide the UAV toward goal-reaching behavior while discouraging collisions, idleness, and unstable flight. The reward is computed at each time step based on distance progress, positional accuracy, motion smoothness, and safety constraints.

Sparse Terminal Rewards At the end of each episode, the UAV receives one of the following terminal rewards:

- Goal-reaching reward: A positive reward of +10 is given if the drone reaches the goal within a threshold distance.
- Collision penalty: A negative reward (e.g., -5) is applied if the UAV crashes into an obstacle or enters a restricted safety zone.
- Out-of-bounds penalty: A penalty is also applied if the UAV exits the simulation bounds.

Continuous Rewards During each episode, the agent is provided with dense feedback based on several continuous reward components:

1. Distance Differential Reward r_d : Encourages the drone to reduce its horizontal and vertical distance to the goal between consecutive steps. It is computed as:

$$r_d = \eta_r \cdot (d_{t-1} - d_t) + \eta_z \cdot (z_{t-1} - z_t) \quad (4.3)$$

where d_t and z_t are the 2D ground-plane and vertical distances to the goal at time t , and η_r, η_z are scaling factors.

2. Pose Error Penalty r_p : Penalizes deviation from the straight-line trajectory connecting the start and goal. The horizontal and vertical errors are clipped and scaled:

$$r_p = \text{clip}\left(\frac{d_{xy}}{R}, 0, 1\right) + \text{clip}\left(\frac{|z - z_g|}{Z_{\max}}, 0, 1\right) \quad (4.4)$$

3. Obstacle Proximity Penalty r_o : If the minimum distance d_o to any obstacle is smaller than the safety threshold d_s , a penalty is computed:

$$r_o = \begin{cases} 1 - \text{clip}\left(\frac{d_o - d_c}{d_s - d_c}, 0, 1\right) & \text{if } d_o < d_s \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

where d_c is the crash threshold and d_s is the safety margin.

4. Action Smoothness and Yaw Penalty r_a : To discourage excessive angular velocity or vertical thrashing, we penalize yaw rate and vertical speed magnitude:

$$r_a = \beta_\omega \cdot \left(\frac{|\omega|}{\omega_{\max}}\right)^2 + \beta_z \cdot \left(\frac{|v_z|}{v_{z,\max}}\right)^2 \quad (4.6)$$

5. Inactivity Penalty r_s : If the drone fails to move meaningfully over a sequence of time steps (defined by velocity and angular thresholds), a fixed penalty of -1 is applied.

The total reward r_t at time t is then aggregated as:

$$r_t = r_d - \lambda_p \cdot r_p - \lambda_o \cdot r_o - \lambda_a \cdot r_a + r_s \quad (4.7)$$

where λ_p , λ_o , and λ_a are weights controlling the influence of each penalty term.

This reward function is crafted to balance exploitation and safety. The distance-based reward components encourage goal-seeking behavior, while the penalties prevent unsafe or inefficient strategies. The inactivity detection mechanism discourages stagnation (e.g., stuck in a dead-end or hovering without progress). Reward shaping is critical to guide exploration in environments with dense obstacles and delayed success feedback. Overall, this hybrid reward design improves convergence and policy robustness across increasingly complex scenarios.

4.2.2 Neural Network Architecture

a. Basic Network Architecture

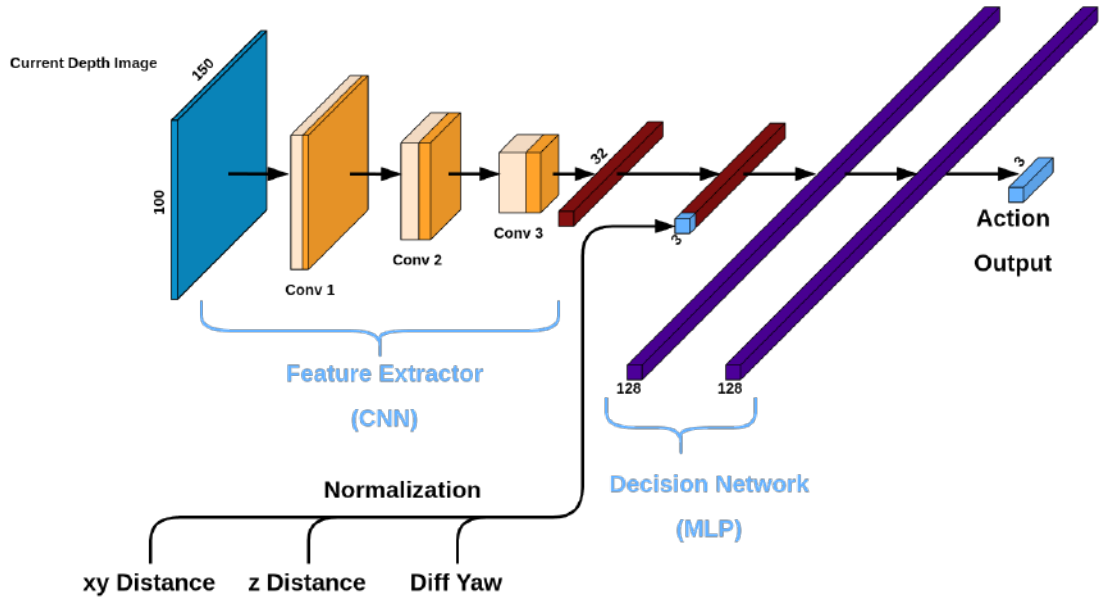


Figure 4.3: Architecture of the Basic Network

Our neural network architecture, illustrated in Figure 4.4, leverages multimodal inputs consisting of visual depth information and relative goal-state variables. To robustly extract features from the visual data, we adopt a pretrained ResNet8 convolutional encoder provided by NTNU [26], specifically optimized for aerial robotics applications involving

depth imagery. The basic network architecture contains 2 main components, the feature extractor and the decision network which are described in Figure 4.3.

Pretrained ResNet8 Feature Extractor. The depth images with a resolution of 100×150 pixels are processed by the ResNet8 convolutional encoder, pretrained by NTNU [26]. This model comprises multiple convolutional layers interspersed with residual (skip) connections to improve gradient flow and learning stability. The detailed structure of the feature extractor is outlined in Table 4.1, where the final dense layer outputs a compact latent feature vector of size 32, corresponding to a latent dimension of 16 (producing both mean and variance vectors).

Table 4.1: Pretrained NTNU ResNet8 Feature Extractor Architecture

Layer Name	Input Dimensions	Kernel Size	Stride	Output Dimensions
Conv2D (conv0)	$1 \times 100 \times 150$	5×5	2	$32 \times 50 \times 75$
Conv2D (conv0_1)	$32 \times 50 \times 75$	3×3	2	$32 \times 25 \times 38$
Conv2D (conv1_0)	$32 \times 25 \times 38$	5×5	2	$32 \times 11 \times 17$
Conv2D (conv1_1)	$32 \times 11 \times 17$	3×3	1	$64 \times 11 \times 17$
Conv2D (conv0_jump_2)	$32 \times 25 \times 38$	4×4	2	$64 \times 11 \times 17$
Residual Sum	$64 \times 11 \times 17 + 64 \times 11 \times 17$	-	-	$64 \times 11 \times 17$
Conv2D (conv2_0)	$64 \times 11 \times 17$	5×5	2	$64 \times 6 \times 9$
Conv2D (conv2_1)	$64 \times 6 \times 9$	3×3	2	$128 \times 3 \times 4$
Conv2D (conv1_jump_3)	$64 \times 11 \times 17$	5×5	4	$128 \times 3 \times 4$
Residual Sum	$128 \times 3 \times 4 + 128 \times 3 \times 4$	-	-	$128 \times 3 \times 4$
Conv2D (conv3_0)	$128 \times 3 \times 4$	5×5	2	$128 \times 1 \times 1$
Flatten	$128 \times 1 \times 1$	-	-	128
Fully Connected (dense0)	128	-	-	512
Fully Connected (dense1)	512	-	-	$32 (2 \times \text{latent dim} = 16)$

Fully Connected Layers. The resulting 32-dimensional latent visual features are concatenated with a low-dimensional state vector $[d_x, y, d_z, \Delta\psi]$, which encodes the UAV’s relative horizontal distance, vertical displacement, and yaw difference to the goal. Prior to concatenation, this state vector undergoes min-max normalization to ensure balanced numerical scales between visual and state inputs. The concatenated feature vector is subsequently fed through two fully connected layers (MLPs) of 64 units each, employing ReLU activations. This dense encoding integrates visual-spatial features with the navigational state, preparing a meaningful representation for temporal reasoning.

Policy and Value Output Heads. Finally, the LSTM’s output is passed into policy-specific heads tailored for the PPO and TD3 algorithms. The PPO architecture outputs a stochastic Gaussian distribution parameterized by mean and variance, whereas the TD3

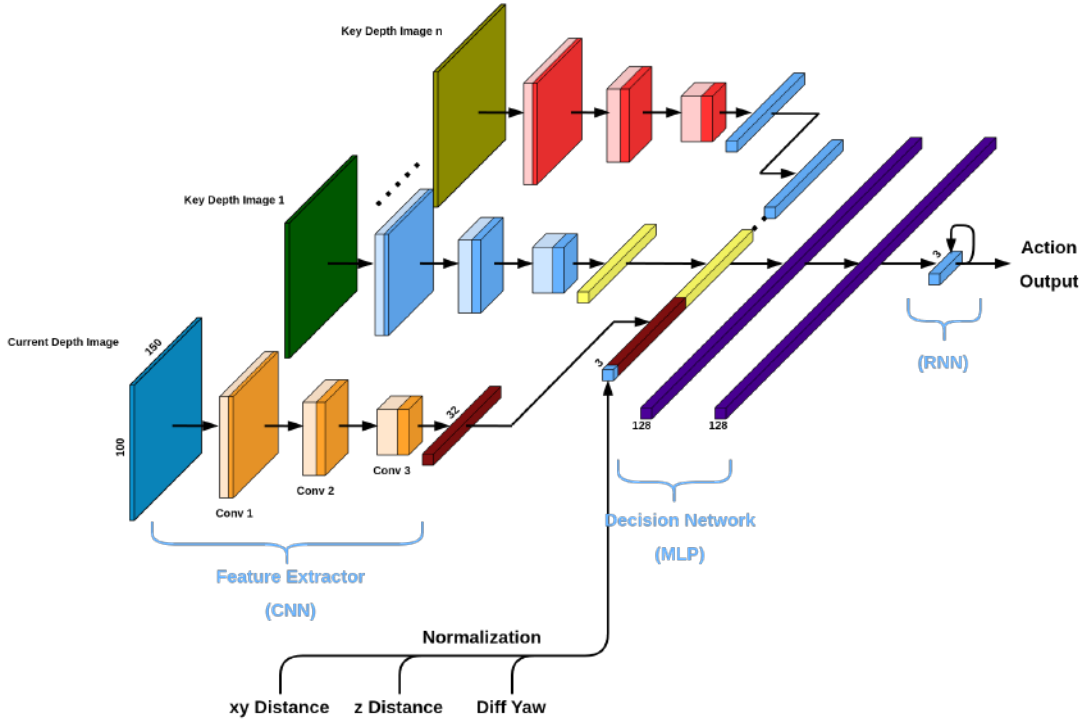


Figure 4.4: Architecture of the Memory-based Network

architecture provides deterministic actions accompanied by dual critic networks for Q-value estimation.

The use of pretrained convolutional features from the NTNU model provides robust visual representations, improving policy stability and enabling more efficient training. The subsequent FCN and recurrent processing allow the network to integrate visual context with goal-oriented information, essential for navigating complex, partially observable environments typical of UAV obstacle avoidance scenarios.

b. Memory-based Network Architecture

The second proposed architecture is a memory-based network architecture. Based on the basic architecture as mentioned above, the memory-based network is added Recurrent NN and Image buffer, which can deal with complexity obstacles as well as partial observability which is a big challenge in reinforcement learning.

Recurrent Module. To explicitly model temporal dependencies and resolve partial observability, the encoded features pass through a single-layer Long Short-Term Memory (LSTM) recurrent unit with 32 hidden units. The LSTM retains historical action

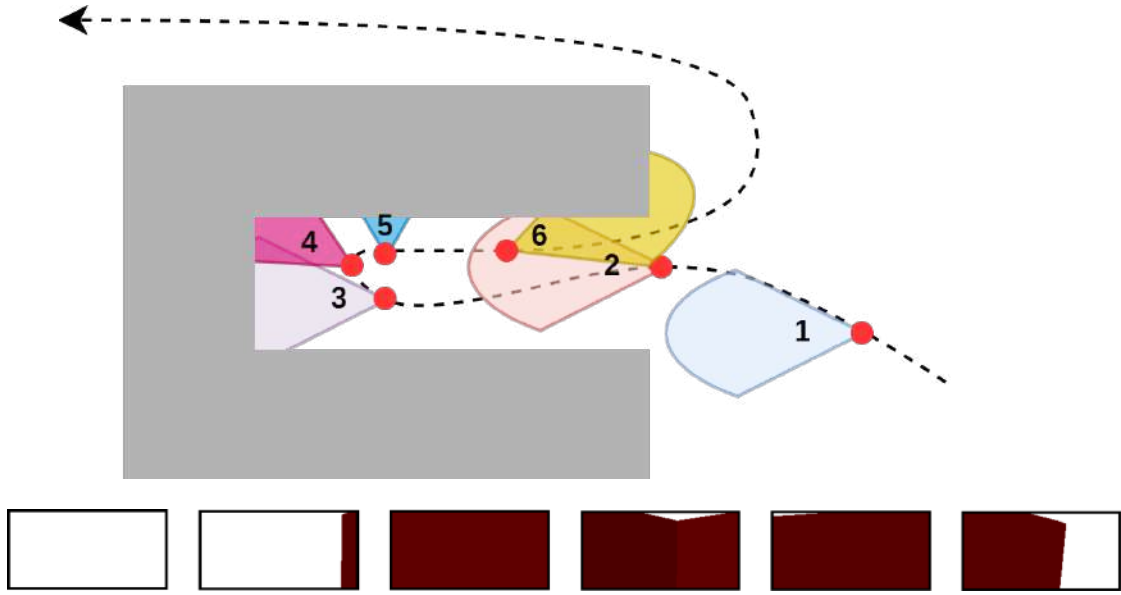


Figure 4.5: Dead-end Case Example with depth images captured at the position 1, 2, 3, 4, 5, 6 from the left to the right.

information critical for navigating environments.

Image Buffer. Instead of using only one depth image to determine a good action for the obstacle avoidance task, we created an image buffer that stores the last number of images. Those images help the drone address some complex scenarios with dead-end obstacles. For example, with a dead-end obstacle in Figure 4.5, the drone can not escape from that scenario with only current depth image at the position 3 or 4. It will gradually decrease velocity, get stuck in that obstacle, and might crash. Basically, the basic network architecture allows the drone to turn left or right and go up or down when there is an obstacle in FoV. It leads to a stuck case when the whole FoV is obstacles. However, intuitively, with the last 4 images and current image, the drone can detect that it is just going into a dead-end obstacle and it should come back to escape and go another way to the goal.

This proposed architecture will help the drone to do reasoning and address complex scenarios effectively and enhance success accuracy. Although a memory based latent space was introduced in [2], which uses LSTM with depth images as input to add memory to their architecture. Our proposed method is simpler and lower computation while still deals with some complex scenarios.

5 Experiments and Results

With the methodology and framework established, we now present the precise configuration details that enable a robust evaluation. This section describes the technical setup, software parameters, and specific scenario design choices that underpin our experiments. Finally, the results of comparison between the methodologies and architectures are discussed at the end of this section.

5.1 Experiments Setup

Reinforcement learning (RL) requires a high-volume, interactive environment in which an agent can repeatedly explore, act, and receive feedback in the form of observations and rewards. For tasks involving mobile robots or aerial drones, this environment must simulate realistic physics, support continuous control, and allow for safe failure during training. A suitable simulation platform should therefore satisfy several key requirements:

- Real-time or accelerated step-wise simulation, allowing millions of training steps within a practical timeframe.
- Continuous action and observation spaces that accurately model UAV dynamics, sensor noise, and control lag.
- Customizability, enabling the placement of obstacles, specification of goal locations, and construction of scenario-rich environments.
- Compatibility with deep RL frameworks, ideally through a Gym-compliant API that integrates seamlessly with PyTorch or TensorFlow.
- Low computational overhead, especially important when training DRL agents for

long durations on limited hardware.

To meet these criteria, we chose the **Gym-PyBullet-Drones** environment [27]. It is an open-source Python simulation suite built upon PyBullet, tailored specifically for multirotor control research. It offers both high-fidelity and lightweight simulation backends and has been widely adopted for research in reinforcement learning for drones, trajectory tracking, and control benchmarking.

To create a meaningful and scalable obstacle avoidance task, we designed a modular simulation environment in Gym-PyBullet-Drones featuring randomized obstacle configurations between the drone’s starting point and a fixed-distance goal. The agent is required to navigate through varying obstacle layouts and reach the goal without collisions, relying solely on onboard depth sensing and relative goal information.

Gym-PyBullet-Drones models quadrotor UAV dynamics using a physics engine that includes aerodynamic drag, mass distribution, propeller thrust/torque characteristics, and contact forces. It exposes both low-level motor RPM control and higher-level velocity or attitude interfaces, making it suitable for training policies across various levels of abstraction. The simulator is also Gym-compliant, meaning it provides a standardized interface (i.e., `step`, `reset`, `observation_space`, `action_space`) which enables easy integration with DRL training frameworks.

5.1.1 Drone Configuration

In this research, we use Crazyflie 2.1 model with parameters in the table 5.1.

Table 5.1: Drone Configuration

Category	Parameter	Value
Dynamics	Frame Size	65 x 65 x 30 mm
	x-axis velocity range	$[-0.5, 0.5]$ m/s
	z-axis velocity range	$[-0.2, 0.2]$ m/s
	yaw velocity range	$[-30.0, 30.0]$ deg/s
	Dynamic frequency	240 Hz
	Control frequency	30 Hz
Depth Camera	Frame Size	100 x 150 pixels
	Frequency	30 Hz
	Field of View	60 deg

5.1.2 Start and Goal Placement

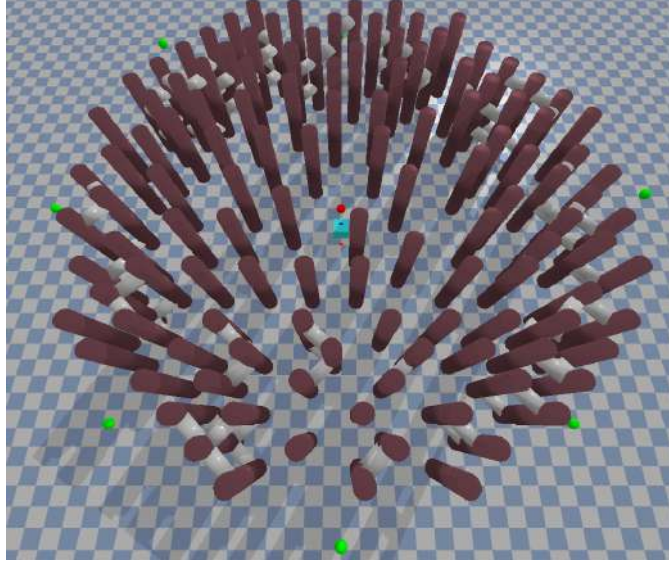


Figure 5.1: Map setup: Red point at the center of the map: drone starting point; Green point: goal which randomly sampled around the map

Table 5.2: Environment Parameters

Category	Parameter	Value
Environment	x range	$[-14, 14]$ m
	y range	$[-14, 14]$ m
	z range	$[0.1, 6]$ m
	Start position	$[0, 0, 3]$ m
	Goal distance	12 m
	Goal height	$[1.5, 4.5]$ m
	XY Accept radius	1 m
	Z Accept radius	0.2 m

Each episode begins with the UAV positioned at the center of a map at a fixed altitude and orientation $(x, y, z, \psi) = (0, 0, h, 0)$, where $\psi = 0$ indicates zero yaw offset. The target goal is placed at a constant radial distance from the origin, randomly sampled along a circular arc in Figure 5.1. So the drone needs to do obstacle avoidance to reach the goal. This design ensures consistent episode length and directional variability, enhance domain randomization while keeping total task difficulty bounded.

5.1.3 Obstacle Design

To challenge the navigation policy, a variety of obstacle primitives are placed between the drone and the goal. Those obstacles illustrates in Figure 5.2, which include:

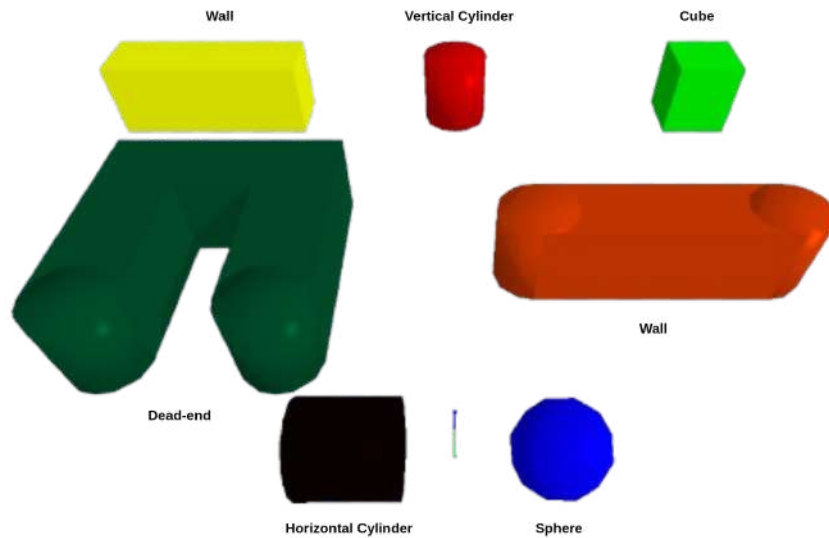


Figure 5.2: Static obstacle types: blocks, walls, spheres, and a U-shaped dead-end.

- Vertical cuboids (walls) and horizontal cuboids (platforms);
- Cylinders and spheres with varying radiusz;
- Blocks with variable height and width;
- A specially constructed U-shaped obstacle representing a *dead-end trap*, requiring backtracking behavior.

These obstacles are randomly positioned within a corridor region between the start and goal mentioned in the last subsection 5.1.2, ensuring that no straight-line path exists but that feasible paths remain. An example obstacle configuration is shown in Figure 5.2.

5.1.4 Low-Level Controller

The outputs of neural network are high-level velocity and orientation commands, which are then translated into rotor speeds by the UAV's built-in low-level controller. As men-

tioned in section 4.1, at each time step, the agent outputs a 3-dimensional action vector:

$$a_t = \begin{bmatrix} v_x^b \\ v_z^b \\ \omega^b \end{bmatrix} \quad (5.1)$$

These outputs are sent to a low-level PID controller provided by the Gym-PyBullet-Drones simulation framework, which maps high-level commands to pulse-width modulation (PWM) signals and rotor RPMs. This separation allows the reinforcement learning agent to focus on trajectory-level decisions (e.g., turning, accelerating, or ascending) without needing to learn low-level motor dynamics.

5.1.5 Curriculum Learning

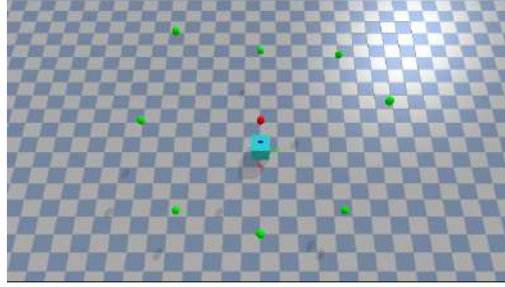
A curriculum learning framework is set up that gradually complexifies the environment by adding more objects to it as the agent learns to successfully navigate the robot in easier environments (with success rate greater than 90% for level 1, 80% for level 2, 75% for level 3, 70% for level 4), while making the environment easier if robots are crashing at a rate above a given threshold (here 5%). We set up a curriculum that involves logging the runs of the robots to measure successes, crashes and episode timeouts.

Let $\mathcal{L} = \{1, 2, 3, 4, 5\}$ be the set of map levels in the curriculum, where S_l is the number of successful episodes and N_l is the total number of episodes at level l . Let T_l denote the success threshold required at level l (Level 1: 90%, Level 2: 85%, Level 3: 80%, Level 4: 75%). The curriculum success rate is defined as:

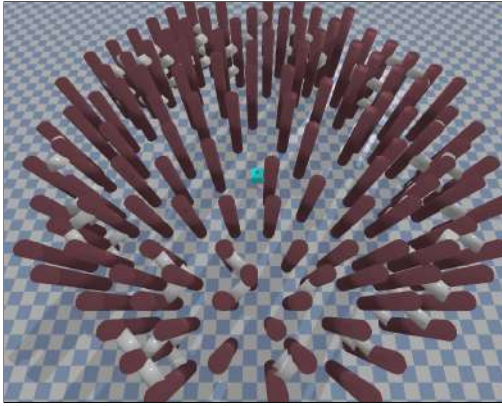
$$\text{SuccessRate}_{\text{curr}} = \frac{\sum_{l \in \mathcal{L}} w_l \cdot \min\left(\frac{S_l}{T_l}, 1\right)}{\sum_{l \in \mathcal{L}} w_l} \quad (5.2)$$

where w_l is the weight assigned to each level as: (1, 2, 3, 4, 5).

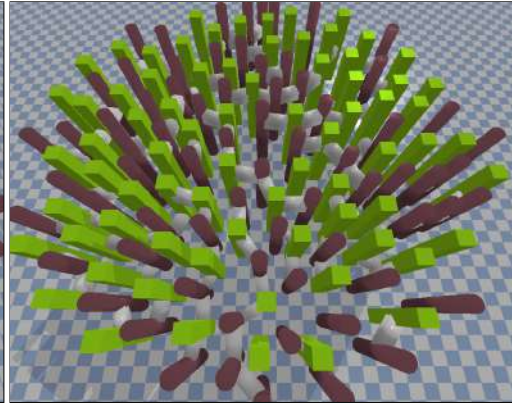
Meanwhile, in terms of the total reward function for curriculum learning, the parameters of the reward function are linearly varied as the curriculum level increases such that $\eta_r^n = \eta_r \theta_l$, where η_r^n is the value of η_r at the curriculum level n , while η_r are positive con-



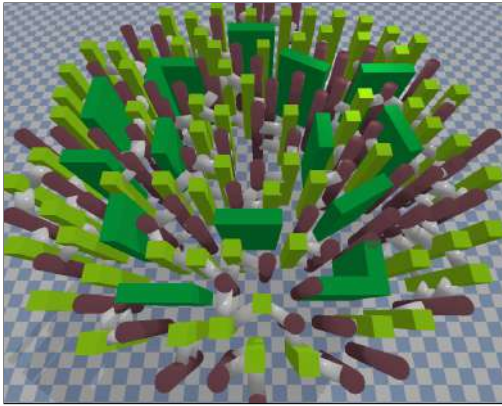
((a)) Map level 1



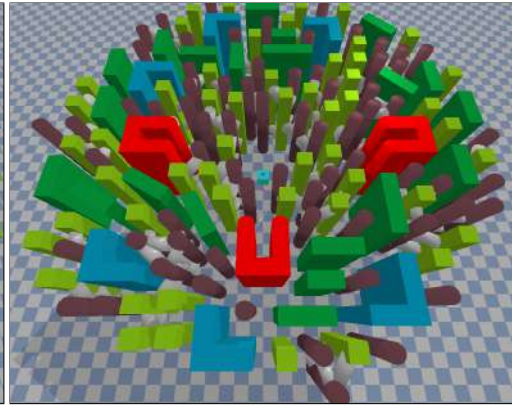
((b)) Map level 2



((c)) Map level 3



((d)) Map level 4



((e)) Map level 5

Figure 5.3: 5 levels of map complexity. (a) Map level 1: empty map, red point is the drone starting point, green points are the goals; (b) Map level 2: add vertical and horizontal cylinders; (c) Map level 3: add vertical cubes; (d) Map level 4: add walls and corners; (e) Map level 5: add dead-end obstacles (U shape).

stants presented in Table 5.3, the weights θ_i at the level 1, 2, 3, 4, 5 are 1.0, 1.2, 1.4, 1.6, 1.8 respectively.

Table 5.3: Reward Function Parameters and Values

Category	Parameter	Value
Reward Function	Goal-reaching reward	+100
	Crash penalty	-120
	Out-of-bounds penalty	-100
	Horizontal distance coefficient η_r	15
	Vertical distance coefficient η_z	75
	Position penalty weight λ_p	0.05
	Obstacle penalty weight λ_o	0.5
	Action smoothness penalty λ_a	0.1
	Yaw error penalty	0.2
	Inactivity penalty	-1
	Safety margin distance d_s	0.40 m
	Crash threshold distance d_c	0.20 m
	Horizontal error scale	1.0
	Vertical error scale	3.0
	Obstacle proximity penalty scale	0.25

The complexity of a map is evaluated on the basis of the density of obstacles in the map space and the type of obstacles added to the map. In this study, we evaluate the complexity of obstacles with increasing complexity as follows: cylinder, cube, wall, corner, and dead-end. Based on that, we defined 5 levels of map complexity described in Figure 5.3. The higher level contains new more complexity obstacles and higher obstacle density in the map. The first level, map level 1 helps training the behavior navigating from the starting point to the goal. The obstacle avoidance behavior is trained with the following levels. At each level, there are 3 training maps randomly chosen in the training process. Examples of each level are presented in the Figure 5.3.

The testing was performed on 3 completely new maps that had not been used for training. 100 goals are randomly selected to determine the success rate of the algorithm.

5.1.6 Noise Injection for Realism and Robustness

To reduce the sim-to-real gap and improve policy robustness, several types of noise and disturbances were systematically injected into the simulation:

- Sensor noise: Gaussian noise was added to each pixel of the depth images gen-

erated by the onboard simulated camera. The noise model is Gaussian with the standard deviation in Table 5.4.

- **State Estimation Noise:** To reflect typical IMU and odometry errors, zero-mean Gaussian noise was added to the position and orientation states.
- **Motor noise:** After each control command (velocity and yaw rate) issued by the policy network was converted to the RPM of four motors by the low-level controller, a Gaussian motor noise was applied to.

Table 5.4: Noise Parameters

Category	Parameter	Value
Noise	Sensor noise standard deviation σ_i	0.03
	State estimation position noise std deviation σ_{pos}	0.02
	State estimation heading noise std deviation σ_ψ	1.5
	Motor RPM noise standard deviation σ_{rpm}	200

By incorporating these quantified noise sources during training, the learned DRL policies were forced to develop strategies that are more robust to uncertainty, lag, and measurement errors—factors which are unavoidable in practical UAV deployments. The addition of noise increased training difficulty, but was crucial for achieving the high success rates observed under varied and randomized test conditions.

5.1.7 Reinforcement Learning Parameters

In this research, we use Stable Baseline 3 library for reinforcement learning PPO and TD3. The parameter settings for training are showed in Table 5.5:

Table 5.5: Training Parameters

Category	Parameter	Value
Training	Discount factor γ	0.99
	Learning rate α	3e-3 (exponential)
	Batch size	64
	Number of environments	12
	Total timesteps	1500000
	Max episode steps	2400

5.2 Results

To validate the comprehensive performance of our UAV autonomous navigation framework, we conducted a number of experiments to do 3 main comparisons:

- PPO vs TD3 performance.
- Basic network architecture vs our proposed architecture with memory and temporal buffer.
- Our proposed deep reinforcement learning method vs a conventional method: EGO-Planner-v2.

5.2.1 PPO vs. TD3 with Basic Network Architecture

We first evaluated both PPO and TD3 using the basic network architecture, with neither LSTM memory nor depth image buffer (only the current frame was provided). Figure 5.5 compares the training progress of the two methods. TD3 consistently achieved higher average episode accuracy and converged more rapidly during early training, thanks to experience replay feature of the off-policy method. However, PPO demonstrated greater training stability, exhibiting less performance variance and fewer abrupt drops during training. After about $5e5$ timesteps, the best PPO models reached an average accuracy of about 85%, while TD3 peaked higher (88%) but with greater fluctuation.

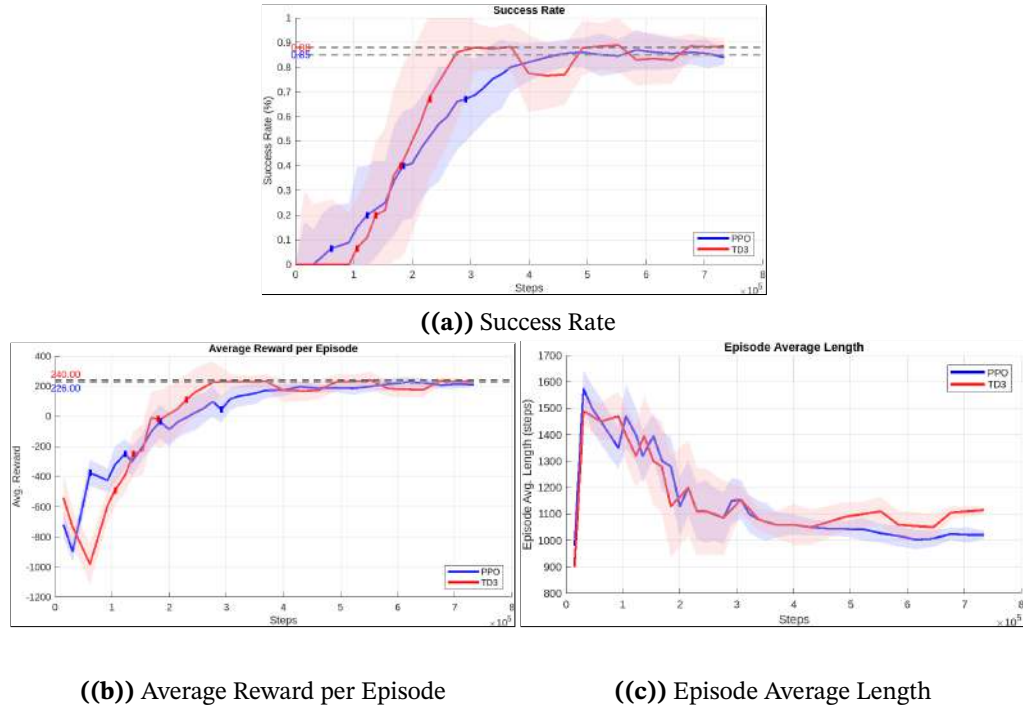


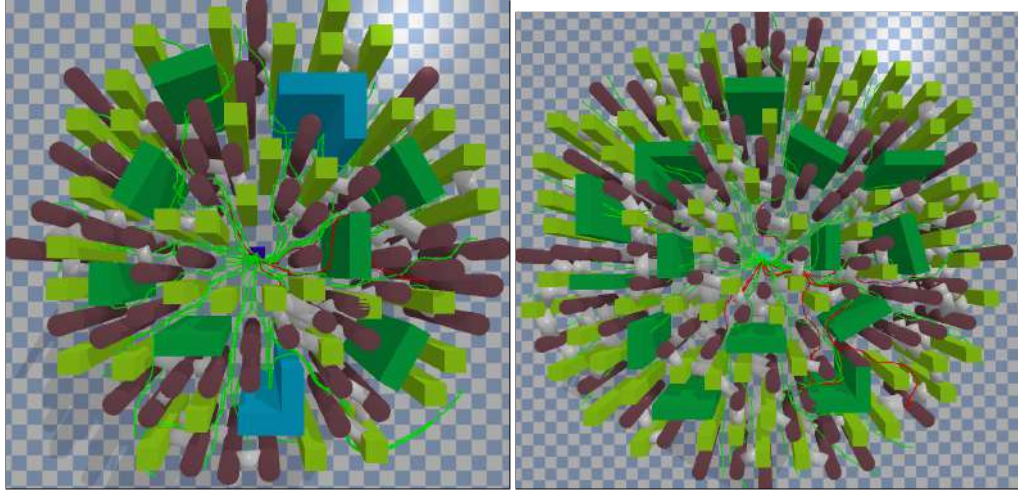
Figure 5.4: PPO vs. TD3 with Basic Network Architecture Result Comparisons.

The blue and red dashed symbols on the graph mark the transition complexity level in the curriculum training process. It is obvious to see that when going to next level, at the beginning the success rate increases slowly, accumulated reward and episode length tend to decrease because of more collision in more complex map. However, after that, the success rate keep increasing until reaching the condition for the next level.

Table 5.6 and Figure 5.5 shows results of the basic architecture in the level 4 training and the level 4 test map. While in the training map, the success accuracy is about 91%, slightly greater than this percentage in the test map, around 89%. Note that this success rate is for a level 4 map instead of the success rate for the curriculum learning during training mentioned in Section 5.1.5. Therefore the achieved success rate is higher than the value in figure 5.4.

Table 5.6: Basic architecture performance in training and testing map.

Environment	Metric		
	AER	AEL	SR
Training Env	274.32	1075.0	91.213%
Testing Env	248.94	1011.0	88.667%



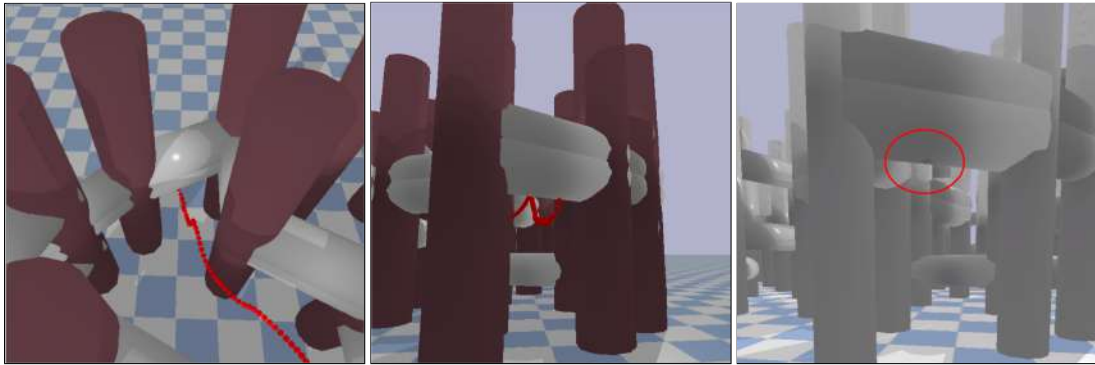
((a)) Train map level 4

((b)) Test map level 4

Figure 5.5: Drone path with PPO on the train and test map. Green path: Path successfully reaches the goal; Red path: Path collides

5.2.2 Effect of Temporal Features: Memory-Based Architecture

Qualitative evaluation and trajectory analysis revealed that both PPO and TD3 policies suffered from two critical failure modes:



((a)) Top view

((b)) Side view

((c)) Crash image

Figure 5.6: Overhead and underfoot obstacles crash case.

1. **Overhead and Underfoot Obstacle Blind Spot:** When the drone maneuvered downward to avoid a low obstacle, previously overhead obstacles moved out of the field of view (FoV) of the camera. As a result, the drone collided with obstacles above it, which it could no longer detect in subsequent frames in Figure 5.6.
2. **Dead-End Traps:** In environments with U-shaped or dead-end obstacles, the drone

frequently entered cul-de-sacs and failed to escape, leading to repeated collisions. The policies lacked memory and could not infer the need to backtrack after entering a trap in Figure 5.7.

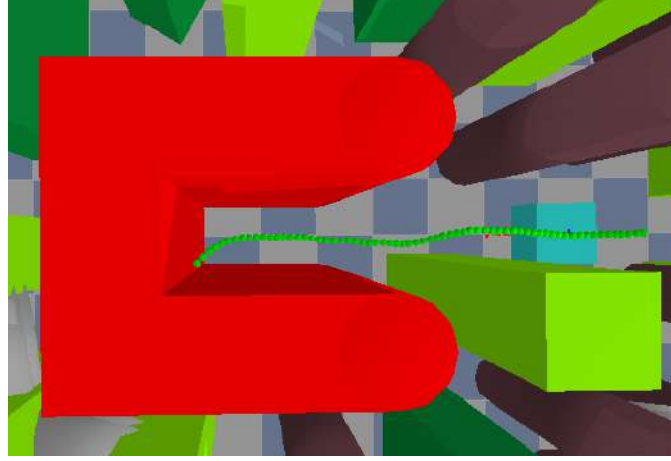


Figure 5.7: Dead-End Traps. The green path is the drone path when it goes into a dead-end obstacle with U-shape and it gets collision at the end of the path.

Due to these limitations, the highest average accuracy plateaued at around 85%, and policy robustness in complex maps was poor.

To address the observed failure cases, we introduced the memory-based architecture in Section 4.2.2 with two improvements: (1) integrating an LSTM module for temporal memory, and (2) fusing the last N key depth images as input (depth buffer). With these modifications, the network learned to leverage historical context—remembering recently seen obstacles and escape trajectories.

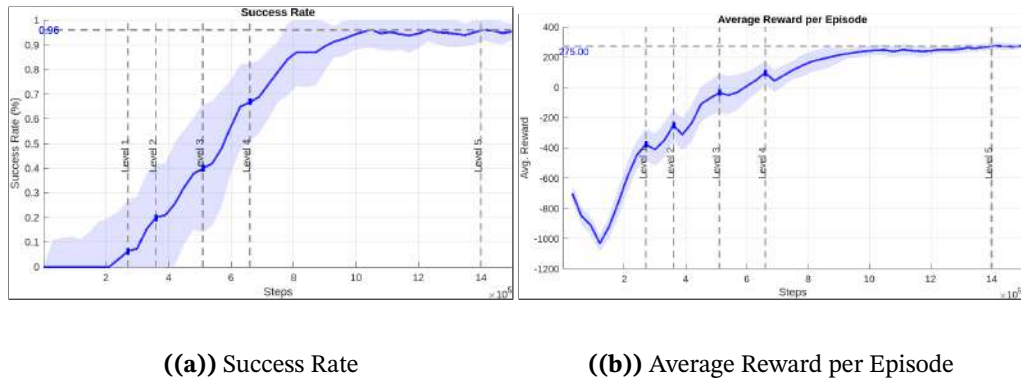


Figure 5.8: Training progress of the memory-based architecture.

As shown in Figure 5.8, the average accuracy increased to **96%**, with markedly im-

proved resilience in both overhead-obstacle and dead-end scenarios. The agent could now anticipate traps, avoid previously observed hazards, and backtrack from dead ends, demonstrating significant gains in generalization and robustness.

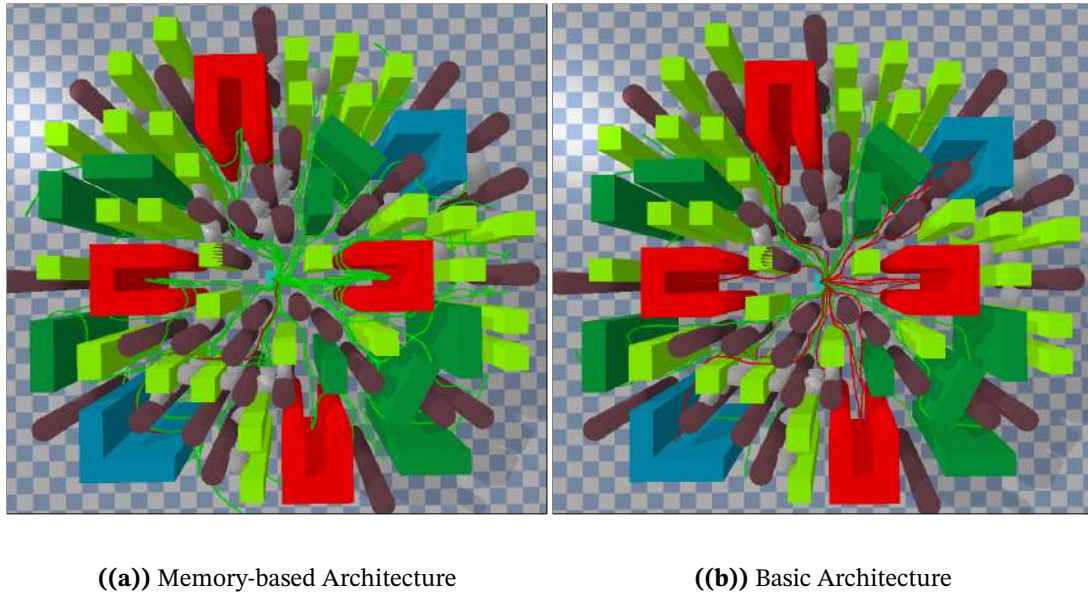


Figure 5.9: Drone path in complex environment (train map level 5). Green path: Path successfully reaches the goal; Red path: Path collides

Obviously, in the Figure 5.9, with the memory-based architecture, the drone can escape from dead-end obstacles and reach the goal easily while the basic architecture cannot deal with it, even the corners (the blue obstacles in the images). These 2 architectures were tested on the completely new map, differ to the training maps.

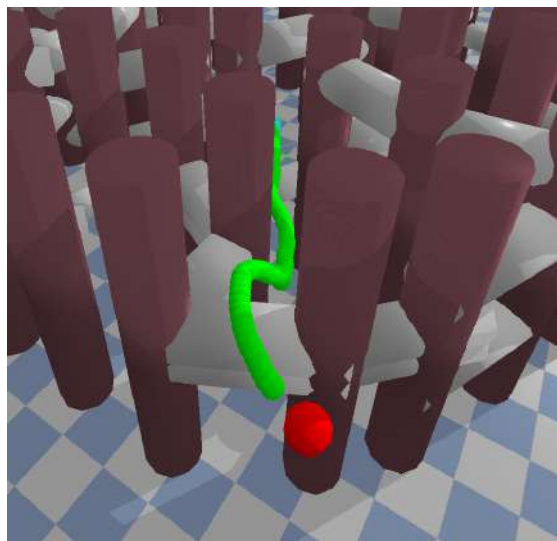


Figure 5.10: The drone path with the underfoot obstacle.

5.2.3 Comparison with Conventional Planning

For a comprehensive assessment, we benchmarked our best-performing DRL policy—specifically, the PPO agent equipped with both LSTM-based memory and a depth image buffer—against EGO-Planner-v2, a state-of-the-art conventional trajectory optimization-based planner. EGO-Planner-v2 constructs kinodynamically feasible, collision-free paths by optimizing over explicit maps, which makes it a strong baseline for UAV obstacle avoidance.

Experiments were conducted on 100 randomly generated maps for both training and testing environments. The performance metrics evaluated were:

- AER (Average Episode Reward)
- AEL (Average Episode Length)
- SR (Success Rate, % episodes reaching the goal without collision)

Table 5.7: Performance Comparison in Training and Testing Environments

Environment	Algorithm	Metric		
		AER	AEL	SR
Training Env	Memory-based DRL	235.62	1125.0	96.667%
	EGO-Planner-v2	210.65	1280.0	91.333%
Testing Env	Memory-based DRL	218.94	1090.0	86.667%
	EGO-Planner-v2	205.86	1350.0	90.0%

From the Table 5.7, the DRL policy slightly outperformed EGO-Planner-v2 in success rate during training (96.7% vs. 91.3%) and achieved a competitive success rate in testing (86.7% vs. 90.0%). Notably, the DRL agent completed episodes with shorter average trajectory length in both settings (1125 vs. 1280 steps in training, 1090 vs. 1350 steps in testing), suggesting more direct navigation and efficient pathfinding.

EGO-Planner-v2 maintained strong overall performance but exhibited a few critical weaknesses in highly randomized or complex environments. The planner’s reliance on accurate, up-to-date maps made it more susceptible to failures in maps with dynamic features or partial observability. In practice, this translated to the planner sometimes committing to suboptimal paths or failing to escape traps, especially in U-shaped or non-convex dead-ends.

These results highlight that a well-designed DRL agent with temporal context can

match or exceed the performance of advanced planning-based pipelines, especially in terms of adaptability and robustness to novel, complex scenarios. The slight difference in the success rate on unseen maps (DRL: 86.7% vs. EGO-Planner-v2: 90.0%) is offset by DRL's significantly improved path efficiency and the resilience to failure modes that traditional planners are prone to.

Ultimately, while conventional planners still hold advantages in interpretability and established safety, DRL-based approaches are closing the gap—offering a scalable and more robust alternative for real-world UAV navigation in cluttered, dynamic environments.

6 Conclusion and Future Work

This thesis set out to address the persistent challenges of autonomous obstacle avoidance for unmanned aerial vehicles (UAVs) using deep reinforcement learning (DRL). Traditional modular pipelines—while interpretable and proven in structured settings—struggle under the realities of cluttered, dynamic environments due to error accumulation, high computational load, and their reliance on hand-crafted features and maps. By proposing an end-to-end DRL-based solution, this work has demonstrated that policies learned directly from raw sensor inputs, coupled with temporal memory, can not only match but in key cases surpass classical planning approaches in both robustness and adaptability.

The experimental results make several points clear. First, policies trained with temporal augmentation—whether through LSTM-based memory modules or via stacked buffers of recent depth images—outperform memoryless baselines. These temporal features allow the agent to maintain a working internal map of the environment, essential for escaping dead-ends, handling occlusions, and recovering from partial observability. The memory-based architecture achieved a 96% success rate on randomized obstacle courses, a significant improvement over both vanilla PPO/TD3 baselines and a state-of-the-art classical planner (EGO-Planner-v2). The framework handled varied obstacles, including overhead and dead-end traps, that typically break reactive or feedforward policies.

Second, the curriculum-based training in the Gym-PyBullet-Drones environment proved critical for generalization. Incrementally increasing obstacle complexity forced the agent to adapt to new situations, making the final policy robust to domain shifts and unseen layouts. The dense, carefully shaped reward function—balancing distance-to-goal, collision penalties, and smoothness constraints—was fundamental to this convergence. Without aggressive reward shaping and penalties for idleness or unsafe maneuvers, the poli-

cies frequently fell into local minima, highlighting the central role of reward engineering in RL for real-world tasks.

However, several limitations remain. Despite outperforming classical planners in the scenarios tested, the DRL policies here are not interpretable in the same way as modular systems; debugging and safety validation remain open problems, especially for deployment in high-risk environments. While the LSTM memory helps with partial observability, it is not a panacea: catastrophic forgetting and failure in highly repetitive or ambiguous layouts were observed. The policies are also highly sensitive to hyperparameters, reward structure, and simulation-to-reality (sim2real) gaps, limiting their direct transfer to real hardware without additional adaptation. The reliance on simulated training environments, while accelerating development and safety, inevitably means that real-world performance will degrade due to noise, dynamics, and sensor uncertainty not modeled in simulation.

Another limitation is the exclusive focus on single-agent navigation. Multi-UAV scenarios introduce a host of new challenges—such as collision avoidance between agents, communication constraints, and non-stationary dynamics—that were outside the scope of this thesis. Similarly, the approach does not yet incorporate high-level mission planning, map building, or semantic understanding. Instead, it is designed for local, reactive navigation and avoidance. These simplifications, while necessary for tractable research, must be addressed in practical systems.

Looking forward, several research directions emerge. First, bridging the sim2real gap is unavoidable for actual deployment. Techniques such as domain randomization, adversarial training, and fine-tuning with real-world data will be essential to make the learned policies robust to unmodeled disturbances, sensor errors, and actuation noise. Second, explainability and safety remain the elephants in the room. Integrating policy distillation, hybrid planning approaches, or rule-based overrides could make DRL-driven navigation more trustworthy and certifiable. Incorporating visual attention or transformer-based memory modules may offer better long-range planning and less susceptibility to LSTM-specific issues like vanishing gradients or catastrophic forgetting.

Third, extending this work to multi-agent settings will require substantial changes,

including explicit coordination mechanisms, decentralized policies, or communication protocols—each with their own scalability and stability challenges. Finally, high-level integration with semantic SLAM (Simultaneous Localization and Mapping), object recognition, and task-level planning could enable end-to-end learning frameworks to not only avoid obstacles but actively pursue complex missions in unknown environments.

In conclusion, this thesis demonstrates that end-to-end deep reinforcement learning, when equipped with temporal reasoning and trained in curriculum-based environments, is a viable and scalable approach for real-time UAV obstacle avoidance. While far from a “solved” problem, this work narrows the gap between academic promise and practical deployment. Future research must now focus on bridging simulation and reality, scaling up to multi-agent settings, and—most importantly—making DRL-based navigation interpretable and verifiable for safe operation in the real world.

References

- [1] M. Kulkarni and K. Alexis, “Reinforcement learning for collision-free flight exploiting deep collision encoding,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 15 781–15 788. <https://doi.org/10.1109/ICRA57147.2024.10610287>
- [2] H. Yu, C. Wagter, and G. C. H. E. de Croon, “Mavrl: Learn to fly in cluttered environments with varying speed,” *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1441–1448, 2025. <https://doi.org/10.1109/LRA.2024.3522778>
- [3] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [4] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. <https://doi.org/10.1109/TSSC.1968.300136>
- [5] S. M. LaValle, “Rapidly-exploring random trees : a new tool for path planning,” *The annual research report*, 1998. [Online]. Available: <https://api.semanticscholar.org/CorpusID:14744621>
- [6] A. T. Stentz, “The d* algorithm for real-time planning of optimal traverses,” Pittsburgh, PA, Tech. Rep. CMU-RI-TR-94-37, October 1994.
- [7] Y. Chen and L. Wang, “Adaptively dynamic rrt*-connect: Path planning for uavs against dynamic obstacles,” 07 2022, pp. 1–7. <https://doi.org/10.1109/CACRE54574.2022.9834188>

- [8] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403–430, 1987.
- [9] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991. <https://doi.org/10.1109/70.88137>
- [10] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997. <https://doi.org/10.1109/100.580977>
- [11] M. Odelga, P. Stegagno, and H. H. Bühlhoff, "Obstacle detection, tracking and avoidance for a teleoperated uav," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 2984–2990.
- [12] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018. <https://doi.org/10.1109/LRA.2018.2795643>
- [13] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge," *Trans. Intell. Transport. Syst.*, vol. 22, no. 1, p. 107–118, Dec. 2020. <https://doi.org/10.1109/TITS.2019.2954952>
- [14] J. Roghair, K. Ko, A. Niaraki, and A. Jannesari, "A vision based deep reinforcement learning algorithm for uav obstacle avoidance," 03 2021. <https://doi.org/10.48550/arXiv.2103.06403>
- [15] X. Han, J. Wang, J. Xue, and Q. Zhang, "Intelligent decision-making for 3-dimensional dynamic obstacle avoidance of uav based on deep reinforcement learning," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1–6. <https://doi.org/10.1109/WCSP.2019.8928110>
- [16] A. Sonny, S. R. Yeduri, and L. R. Cenkeramaddi, "Q-learning-based unmanned

- aerial vehicle path planning with dynamic obstacle avoidance,” *Applied Soft Computing*, vol. 147, p. 110773, 2023. <https://doi.org/https://doi.org/10.1016/j.asoc.2023.110773>
- [17] O. Bouhamed, H. Ghazzai, H. Besbes, and Y. Massoud, “Autonomous uav navigation: A ddpg-based deep reinforcement learning approach,” 03 2020. <https://doi.org/10.1109/ISCAS45731.2020.9181245>
- [18] J. Wang, Z. Yu, D. Zhou, J. Shi, and R. Deng, “Vision-based deep reinforcement learning of unmanned aerial vehicle (uav) autonomous navigation using privileged information,” *Drones*, vol. 8, no. 12, 2024. <https://doi.org/10.3390/drones8120782>
- [19] I. Kim, S. Shin, J. Wu, S. D. Kim, and C. G. Kim, “Obstacle avoidance path planning for uav using reinforcement learning under simulated environment,” in *Proceedings of the IASER 3rd International Conference on Electronics, Electrical Engineering, Computer Science*, Sapporo, Japan, 2017, pp. 34–36.
- [20] W. Jiang, T. Cai, G. Xu, and Y. Wang, “Autonomous obstacle avoidance and target tracking of uav: Transformer for observation sequence in reinforcement learning,” *Knowledge-Based Systems*, vol. 290, p. 111604, 2024. <https://doi.org/10.1016/j.knosys.2024.111604>
- [21] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” 2017. [Online]. Available: <https://arxiv.org/abs/1502.05477>
- [22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *ArXiv*, vol. abs/1707.06347, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:28695052>
- [23] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” vol. 32, no. 1, pp. 387–395, 22–24 Jun 2014. [Online]. Available: <https://proceedings.mlr.press/v32/silver14.html>
- [24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2019. [Online]. Available: <https://arxiv.org/abs/1509.02971>

- [25] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” 2018. [Online]. Available: <https://arxiv.org/abs/1802.09477>
- [26] M. Kulkarni and K. Alexis, “Task-driven compression for collision encoding based on depth images,” in *Advances in Visual Computing: 18th International Symposium, ISVC 2023*, ser. Lecture Notes in Computer Science, G. Bebis, G. Ghiasi, Y. Fang, A. Sharf, Y. Dong, C. Weaver, Z. Leo, J. J. L. Jr., and L. Kohli, Eds., vol. 14302. Springer Nature Switzerland, 2023, pp. 259–273. https://doi.org/10.1007/978-3-031-47966-3_20
- [27] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, “Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7512–7519. <https://doi.org/10.1109/IROS51168.2021.9635857>

7

Deep Reinforcement Learning for Robot Manipulation
Vania Katherine Mulia

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING

MASTER'S THESIS No. 143

Deep Reinforcement Learning for Robot Manipulation

Vania Katherine Mulia

Zagreb, June 2025.

My deepest gratitude goes to my thesis supervisor, Prof. Narcís Palomeras, for all the technical guidance, discussions, and support throughout the course of this thesis.

I would also like to extend my gratitude to the IFRoS programme for the opportunity to pursue a master's degree and learn from the experts in the field of robotics. I would like to acknowledge the coordinators at the University of Girona, Prof. Narcís Palomeras and Prof. Nuno Gracias, the coordinators at the University of Zagreb, Prof. Tamara Petrović and Prof. Stjepan Bogdan, and all the professors in both universities for the knowledge and support throughout this master's programme. I would also like to thank Imma Güell from the IFRoS coordination team for all the assistance during my study.

I would also like to thank my fellow IFRoS students for the bond that we have built over the 2 years of our study. A special thanks goes to Tanakrit and Precious for the help and companionship during this thesis process.

A heartfelt thank you to my mother for continuously being my rock during my studies, despite being very far away. To my two brothers, for supporting my decision to do this master's degree.

And finally, to my dad, who I believe is watching me from heaven... Before I left home, I made a promise in front of you that if the time comes, I will stay strong and keep doing what I need to do. Now, I can finally say that I have made it!

Deep Reinforcement Learning for Robot Manipulation

Abstract

Robotic manipulation continues to be an active area of research due to its broad range of real-world applications. Among its benchmark tasks, the peg-in-hole problem remains particularly challenging, requiring high-precision control under environmental uncertainty. This thesis presents a framework based on Deep Reinforcement Learning (DRL) to train a robotic manipulator to autonomously solve the peg-in-hole task. The proposed approach uses curriculum learning to train a single policy capable of handling all phases of the task: approach, contact-based hole search, and insertion. The curriculum is further extended to incorporate observation noise and force penalization, encouraging the emergence of compliant behaviors during contact. Training is conducted in a custom-designed, physics-based simulation environment. Simulation results demonstrate that the learned policy can complete the peg-in-hole task, though it faces difficulties in balancing task success with compliant interaction. To evaluate the potential for real-world deployment, the trained policy is transferred to a physical robot. Tests reveal several sources of sim-to-real discrepancy, particularly in the modeling of contact dynamics. Nonetheless, partial success in real-world trials suggests the viability of sim-to-real transfer for DRL-trained policies. Overall, this work contributes to the understanding of DRL's capabilities and limitations in solving complex robotic manipulation tasks such as peg-in-hole assembly.

Keywords: Robot manipulation, peg-in-hole task, deep reinforcement learning, curriculum learning, sim-to-real transfer

CONTENTS

1. Introduction	1
1.1. Background	1
1.2. Problem Statement	3
1.3. Thesis Contributions	3
1.4. Structure of Thesis	4
2. Literature Review	5
2.1. Theoretical Foundation of Reinforcement Learning	5
2.2. Literature Survey	6
2.2.1. DRL Algorithms in Robot Manipulation	6
2.2.2. Challenges in Implementation	8
2.3. Curriculum Learning	9
2.4. Gaps in Literature and Research Focus	10
3. Methodology	12
3.1. Deep Reinforcement Learning Framework	12
3.1.1. Task Description	12
3.1.2. Algorithm Used	12
3.1.3. Observation and Action Space	14
3.1.4. Reward Function	16
3.1.5. Domain Randomization	17
3.1.6. Curriculum Implementation	18
3.2. Simulation Setup	20
3.3. Simulation-to-Real Transfer	23
3.3.1. Real-World Experiment Setup	23
3.3.2. Software Architecture	24

4. Results and Discussions	26
4.1. Simulation Results	26
4.1.1. Peg-in-Hole Task Evaluation: Stages 1 to 7	26
4.1.2. Evaluation of Final Stage	29
4.2. Test Results with Real Robot	32
4.2.1. Evaluation of Transferred Policy	33
4.2.2. Simulation-to-Reality Gaps	33
4.3. Insights from Training and Deployment Results	37
5. Conclusion and Future Works	38
Bibliography	39

1. Introduction

1.1. Background

Robotic manipulation has gained substantial attention in research due to its various fields of application. Among the benchmark problems, the peg-in-hole task is a classical challenge in robotics and automation, involving the insertion of a peg into a hole with tight tolerances. This task is commonly divided into two phases: the search phase, where the robot locates the hole and aligns the peg, and the insertion phase, where the peg is inserted while adjusting its orientation relative to the hole. An illustration of the peg-in-hole task can be seen in Figure 1.1. This task requires precise alignment, force control, and often sensory feedback to compensate for uncertainties in the peg's position, orientation, and the hole's location. The problem is widely studied as it has practical applications in manufacturing, assembly, surgical robotics, and other fields requiring high-precision manipulation.

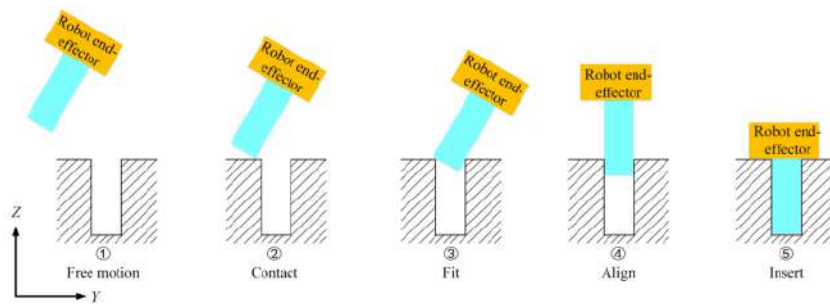


Figure 1.1: Peg-in-Hole Task [1]

The peg-in-hole task presents several challenges, most notably dealing with uncertainties in position and orientation estimation. Uncertainties in hole position estimation hinder the search phase [2, 3, 4]. Furthermore, the algorithm to solve this task also has to consider possible orientation misalignments between the peg and the hole [2, 4]. Small misalignments between the peg and the hole can cause failure, requiring precise sensing and control.

Another challenge that may arise is related to friction and contact forces [4, 5]. As the task is considered a contact-rich manipulation task, unexpected friction or excessive force can lead to jamming, binding, or even damaging the peg or hole. Furthermore, in most robotic systems, the algorithm has to deal with perception limitations, including noise, occlusions, or inaccuracies from visual and/or tactile sensors [3], making it difficult to detect exact positioning. The problem can even be expanded to include possible uneven surfaces [5] and insertion of non-rigid objects [3].

Traditional control solutions for peg-in-hole tasks typically rely on precise motion planning, force control, and compliance strategies. Control strategies can be formulated as position-based (i.e. by defining a trajectory to align the peg with the hole and insert it), force-based (i.e. by using force/torque sensors to detect contact forces and adjust movements accordingly), or a combination of both. Other control strategies include visual servoing and passive compliance control. However, traditional methods struggle with uncertainties (e.g., sensor noise, surface variations). They also require task-environment modeling, precise calibration, and proper controller tuning, leading up to the requirements of significant time, effort, and expertise. These methods are also not robust to uncertainties and lack adaptability to dynamic or changing conditions in real-world applications. All these limitations lead to the use of more modern approaches and techniques, including reinforcement learning.

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. Instead of being explicitly programmed with rules, the agent explores different actions and receives feedback in the form of rewards or penalties, gradually improving its strategy over time. For tasks like peg-in-hole insertion, RL allows a robot to learn through trial and error, adapting to uncertainties in positioning, friction, and contact forces. Instead of relying on predefined rules, it optimizes insertion strategies by learning from interactions and maximizing long-term success rates. In recent years, Deep Reinforcement Learning (DRL)—which combines RL with deep neural networks—has enabled learning in high-dimensional, continuous action spaces, making it suitable for complex robotic tasks involving raw sensor inputs such as vision or force feedback. However, applying DRL to peg-in-hole tasks remains non-trivial. Key challenges include sample inefficiency, difficulty in handling sparse or delayed rewards, poor sim-to-real transfer, and safety concerns during real-world training.

1.2. Problem Statement

This thesis addresses the problem of enabling a robotic manipulator to autonomously learn the peg-in-hole insertion task using DRL. The goal is to develop a learning-based control policy that can generalize across varying hole positions and orientations, cope with real-world uncertainties, and achieve high insertion success rates. A well-trained policy should accomplish the task reliably while respecting physical constraints, such as force limits and the robot's kinematic capabilities. The work involves designing appropriate training environments (both simulated and real), state representations, and reward structures that facilitate learning. This thesis also investigates the feasibility and challenges of simulation-to-reality (sim2real) transfer of a policy trained using DRL in a physics-based simulated environment. Ultimately, the research aims to demonstrate that DRL can be a viable and effective solution for precision assembly tasks in industrial and unstructured settings.

1.3. Thesis Contributions

The main contributions of this thesis are as follows:

- Design and implementation of a custom peg-in-hole task environment in a physics-based simulation environment (MuJoCo), including task-specific observation and action spaces, and a reward function suitable for reinforcement learning.
- Integration of the custom environment with a DRL training pipeline using the Stable-Baselines3 library.
- Development of an efficient learning strategy to solve the peg-in-hole task, which includes both search and insertion phases, while exhibiting a force-compliant behavior.
- Experimental evaluation of the trained policy's ability to generalize across varying hole positions and orientations under uncertainty.
- Investigation of sim-to-real transfer, including domain adaptation challenges and performance evaluation on a real robotic manipulator.

1.4. Structure of Thesis

The remainder of this thesis is organized as follows: Chapter 2 provides the theoretical foundation of DRL and an overview of past works on DRL for robot manipulation. Chapter 3 details the proposed methodology, including the algorithm used and the environment settings in both the simulation and the real world. Chapter 4 presents the results of both simulation and real-world experiments, demonstrating the effectiveness of the proposed approach and challenges found during the experiments. Finally, Chapter 5 concludes the thesis with a conclusion of the findings and suggestions for future research directions.

2. Literature Review

2.1. Theoretical Foundation of Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. Instead of being explicitly programmed with rules, the agent explores different actions and receives feedback in the form of rewards or penalties, gradually improving its strategy over time. At each time step t , the agent observes the current state $s_t \in \mathbb{S}$ (where $\mathbb{S}$ is the state space), takes an action $a_t \in \mathbb{A}$ (where $\mathbb{A}$ is the action space), and receives a reward $r_{t+1} \in \mathbb{R}$ from the environment. The environment then transitions to a new state s_{t+1} based on the transition dynamics $P(s_{t+1}|s_t, a_t)$. It is worth mentioning that the RL problem can be modeled as a Markov Decision Process (MDP). This process can be illustrated in Figure 2.1.

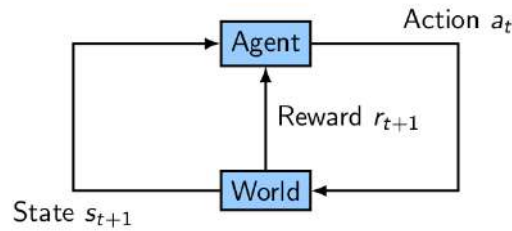


Figure 2.1: Schematic of Reinforcement Learning

The goal of RL is to learn a policy $\pi(s)$ that maps states to optimal actions to maximize the cumulative future reward, not just immediate rewards. This is often formalized using a discounted reward sum, expressed as

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (2.1)$$

where $\gamma \in [0, 1]$ is the discount factor that balances immediate and future rewards.

To achieve this, RL relies on value functions, which are evaluations of states. There are two types of value functions:

- State-value function $V^\pi(s)$ estimates the expected return starting from state s and following policy π .

$$V^\pi(s) = \mathbb{E}_\pi[R_t | s_t = s] \quad (2.2)$$

- Action-value function (also called Q-function) $Q^\pi(s, a)$ estimates the expected return for taking action a in state s and then following policy π .

$$Q^\pi(s, a) = \mathbb{E}_\pi[R_t | s_t = s, a_t = a] \quad (2.3)$$

In many environments, the agent does not get access to the full internal state of the environment. Instead, it receives an observation — a possibly partial or noisy view of that state. The observation is what the agent uses as input to its policy $\pi(o)$, which maps observations to actions. Observations are defined based on the task and environment. For instance, in a robotic manipulation task, the observation might include joint positions, velocities, and force sensor readings.

In more complex environments, Deep Reinforcement Learning (DRL) combines RL with deep neural networks to approximate value functions or policies, enabling the agent to handle high-dimensional state spaces, such as images or complex sensor data. Training is done to optimize the neural network parameters that estimate the value functions. As with any deep learning approach, optimization is done by minimizing a loss function.

2.2. Literature Survey

2.2.1. DRL Algorithms in Robot Manipulation

One of the first DRL algorithms to be applied to robot manipulation tasks is Deep Q-Network (DQN) [6]. The main idea behind this algorithm is to use a neural network to predict Q-values for all actions in a given state and update Q-values using the Bellman equation. DQN has been studied in a wide range of applications, including robot manipulation, due to its simplicity and applicability to scenarios with high-dimensional state space [7, 8]. In [2], DQN is used with Long Short-Term Memory (LSTM) network to train a 7-axis articulated robot arm to perform a high-precision cylindrical peg-in-hole task. In this paper, separate

Q-functions were learned for the search and insertion phases, and the resulting policies are robust to hole position and angle errors. In [5], DQN is used to train an industrial robot to perform an anchor-bolt insertion task, which is similar to the peg-in-hole task, for holes in concrete with variable shape (including chamfered shape of holes) and surface finish. In [9], multi-pass DQN (MP-DQN), which is an improvement to DQN, is used to train a robotic manipulator to perform peg-in-hole tasks with different peg-hole shapes: square, pentagon, and triangle. This paper defines the actions as primitives, each with its own parameters taken as velocities and contact force limits.

As research on DRL evolves, more DRL algorithms has been introduced and applied to robot manipulation tasks. A key limitation of DQN is that it cannot be applied to continuous action spaces. To address this limitation, Deep Deterministic Policy Gradient (DDPG) [10] is an off-policy actor-critic DRL algorithm that extends DQN to continuous action spaces using an actor-critic framework. In [3], DDPG is used to train a robot arm to perform narrow-clearance peg-insertion and deformable clip-insertion tasks. DDPG is further improved to Twin Delayed DDPG (TD3) [11], which adds several components to avoid overestimation of Q-values. Applications of TD3 in robot manipulation tasks can be seen in [12, 13].

Other DRL algorithms, such as Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO), can also be found in studies of robot manipulation [7]. TRPO [14] is a policy-based DRL algorithm that provides stable and theoretically grounded updates, suitable for complex manipulation tasks. An implementation of TRPO can be found in [15]. PPO [16] is a modification of TRPO, and it also gains favor in robotic manipulation due to its ability to learn from smaller amounts of data without overfitting. An implementation of PPO can be found in [17, 18]. However, both TRPO and PPO tend to be data inefficient and produce high variance, making them less popular in robotic manipulation.

Soft Actor-Critic (SAC) [19] is one of the newer off-policy algorithms yet highly studied in robot manipulation applications. Different from DDPG, the key idea of SAC is to improve stability and exploration by adding entropy regularization, balancing exploitation and exploration. In [20], SAC in combination with a trajectory generator is used to control the motion of the end-effector to accomplish a narrow-clearance peg-in-hole task with a square-shaped peg and hole. In [21], SAC is used in combination with force controllers (parallel position/force control and admittance control) to perform peg-in-hole tasks. In this paper, in addition to the desired end-effector pose, the learned policies also include the force controller

parameters, eliminating the need for dynamic tuning of the force controller to comply with the force limits.

2.2.2. Challenges in Implementation

Implementing deep reinforcement learning (DRL) in robotic manipulation tasks presents several significant challenges. One of the most critical is the design of an effective reward function that appropriately guides the learning process. Poorly shaped rewards can lead to suboptimal or unintended behaviors. Most studies use continuous reward functions that map the distance or pose error between the end-effector (carrying the peg) and the hole to a scalar reward signal [9, 12, 18]. Some works additionally incorporate penalties based on the number of time steps taken to encourage faster task completion [2, 21], or include terms that penalize excessive contact forces at the end-effector to prevent jamming or damage [4, 13, 21].

Other approaches utilize sparse reward functions with intermediate rewards provided at key task milestones [5, 22]. In rare cases, a purely sparse binary reward function is used, where the agent receives a positive reward only upon task completion and zero or negative otherwise [3, 23]. Due to the difficulty of learning from such sparse signals, these methods are often augmented with techniques like Hindsight Experience Replay (HER) [23] or learning from demonstrations [3] to improve sample efficiency and learning stability. However, applying these techniques to peg-in-hole tasks is not always straightforward. For example, HER can suffer from a goal mismatch problem, as it assumes that any achieved state could be treated as a valid goal. This assumption may not hold in tasks with strict goal constraints, such as those requiring a specific trajectory or insertion orientation, which may lead to misleading or uninformative learning signals.

Another major challenge lies in obtaining appropriate observations. Poorly designed observation spaces can result in inefficient learning or policies that fail to generalize, particularly in real-world settings where sensors are noisy or imperfect. Many studies rely on end-effector force and torque readings from force-torque sensors [5, 22], especially during the contact-rich hole search phase, where such feedback helps infer the relative position of the hole. End-effector pose or pose error is also commonly used, especially in simulation-based training where precise information is readily available [9, 12, 21]. Other frequently used observations include joint positions and velocities [3, 23, 24]. When visual feedback is available, camera input can help locate the hole before contact [3, 18, 25], although image

observations often require additional processing to extract meaningful features.

Finally, transferring policies trained in simulation to real hardware is particularly challenging. Simulations often fail to capture all aspects of real-world dynamics, including sensor noise, unmodeled contact behaviors, and latency, which can degrade real-world performance [9, 12]. To improve generalization, domain randomization is commonly employed during training [26]. The basic idea of domain randomization is to expose the agent to a wide variety of randomized parameters in the simulation, such as object positions, physical properties, lighting conditions, and sensor noise, so that the learned policy becomes robust to the variations it may encounter in the real world. An example of it can be seen in [20]; each learning episode is set with a randomized initial position (or pose) of the peg and/or the hole.

2.3. Curriculum Learning

Curriculum learning is a learning paradigm in which a complex task is decomposed into a sequence of simpler subtasks arranged in increasing order of difficulty (also known as a curriculum). The basic idea is borrowed from the way humans learned – by starting from easier tasks and progressively working their way to mastering more difficult tasks. By leveraging knowledge gained from simpler tasks, exploration in more complex tasks becomes more efficient, accelerating the overall learning process [27].

In general, there are three important aspects when designing a curriculum:

- Task generation – that is, how to decompose the task into simpler tasks. The quality of the sub-tasks in a curriculum determines the quality of the curriculum.
- Sequencing – that is, how to order the sub-tasks such that the agent can benefit from the experience collected from the previous, easier tasks to learn more complex tasks.
- Stopping criteria – that is, how to decide when to stop training with the current sub-task and advance to the next sub-task. The most common approach is to advance to the next sub-task when the performance on the current sub-task has converged. An alternative approach is to train the agent on each sub-task for a fixed number of episodes or timesteps [27].

The majority of previous research has employed manually constructed curricula,

in which the sequence of tasks is chosen by a human. For example, in robotics, an expert might establish a sequence starting from simple tasks like picking up objects and gradually leading to more complex ones like assembling parts. But lately, researchers have started looking into automated curriculum sequencing techniques [27].

Implementation of curriculum in RL has gained popularity in recent years. In [28], the curriculum is defined by varying initial horizontal position offset between the peg and the hole. Since a higher offset means higher difficulty in the hole search phase, the curriculum is designed such that the agent trains with a lower offset before gradually increasing the range of offset in three stages. In this study, the stopping criterion is defined by the average success rate of the last n episodes.

In [29], curriculum learning is combined with domain randomization by modifying the domain randomization parameters based on the difficulty level. This study also compares two different curriculum evolution criteria: linear (i.e. difficulty increases monotonically with the episode count) and adaptive (i.e. adjust difficulty based on success rate of the last few episodes). Furthermore, this study implements dynamic reward function such that the reward is scaled by the difficulty level, thus harder tasks yield higher reward. This encourages the agent to prioritize mastering harder tasks for better real-world transfer.

2.4. Gaps in Literature and Research Focus

Based on the literature survey presented above, this thesis aims to address the following research gaps:

- **Learning a single policy that addresses both search and insertion phases.** Some prior works focus primarily on the search phase, often assuming that the peg and hole are already perfectly aligned in orientation [5, 30]. In contrast, this study removes that assumption and aims to train a policy capable of handling orientation misalignments throughout the task. Additionally, in [2], the peg-in-hole task is addressed by training separate policies for the search and insertion phases. Here, we propose learning a single policy that can seamlessly handle both phases within the same framework.
- **Inclusion of an approach phase.** In most existing studies, the agent is trained under the assumption that the peg starts either directly above

the hole or very close to it in height, often already in contact with the surrounding surface [2, 9, 13, 28]. This thesis introduces an additional approach phase, where the peg begins at a greater distance from the hole. The agent must learn to generate an appropriate trajectory to approach the hole before initiating contact-based search and insertion. This is achieved through curriculum-based training that gradually increases the complexity of the task.

- **Learning force-compliant behavior without relying on compliance controllers.** Many previous works rely on explicit compliance controllers in combination with deep reinforcement learning to achieve compliant behavior during contact [4, 12, 20, 21, 28]. In contrast, this thesis investigates whether a policy can be trained to exhibit force-compliant behavior through learning alone, without the aid of any compliance controller.
- **Learning without the use of demonstrations or prior knowledge.** Several prior studies incorporate demonstrations or prior knowledge to accelerate learning [3, 4, 17, 18]. This thesis adopts a different approach by training the policy purely through interaction with the environment, without relying on demonstrations. This simplifies the learning framework and removes the need for any expert data collection prior to training.

To the best of our knowledge, no existing study has addressed all of the above aspects in a unified framework.

3. Methodology

3.1. Deep Reinforcement Learning Framework

3.1.1. Task Description

The objective of this thesis is to train a single policy to solve the peg-in-hole insertion task that includes approaching the hole, searching the hole in a contact-rich manner, and inserting the peg fully into the hole. The task involves a 6-DoF UR3e robotic manipulator, which is responsible for inserting a cylindrical peg into a fixed hole with tight tolerances. The peg is rigidly attached to the robot's end-effector and must be inserted vertically downward into the hole located on a horizontal surface.

The peg has a circular cross-section with a radius of 22 mm and a length of 100 mm, while the hole has a slightly larger radius of 25 mm. Thus, there is a 3-mm clearance between the peg and the hole. The depth of the hole is set to 60 mm, requiring the robot to insert the peg fully to achieve task success.

The trained policy will be applied to the UR3e manipulator arm equipped with a twist controller. Thus, the actions generated by the policy should exhibit compliant behavior without the need for a compliant controller on top of the trained policy.

3.1.2. Algorithm Used

In this thesis, Soft Actor-Critic (SAC) algorithm is used to train the robot to perform the peg-in-hole task. SAC [19] is an off-policy actor-critic algorithm based on the maximum entropy reinforcement learning framework. Unlike traditional RL methods that aim to maximize expected return, SAC seeks to maximize both return and policy entropy, encouraging exploration and robustness.

The overall objective is to maximize:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} [r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))] \quad (3.1)$$

where $r(s_t, a_t)$ is the reward, $H(\pi(\cdot|s_t)) = -\mathbb{E}_{a_t \sim \pi} [\log \pi(a_t|s_t)]$ is the policy entropy, and α is a temperature parameter that balances the trade-off between exploration (entropy) and exploitation (reward).

Actor-Critic Framework. SAC follows the actor-critic architecture, which consists of two main components: the **actor**, which represents the current policy and selects actions based on the current state, and the **critic**, which estimates the value of those actions using a value function (typically a Q-function). The actor is updated to improve its policy using feedback from the critic, which in turn is trained to predict expected returns more accurately. This division allows for stable and sample-efficient learning, especially in continuous action spaces. The schematic of this framework can be seen in Figure 3.1.

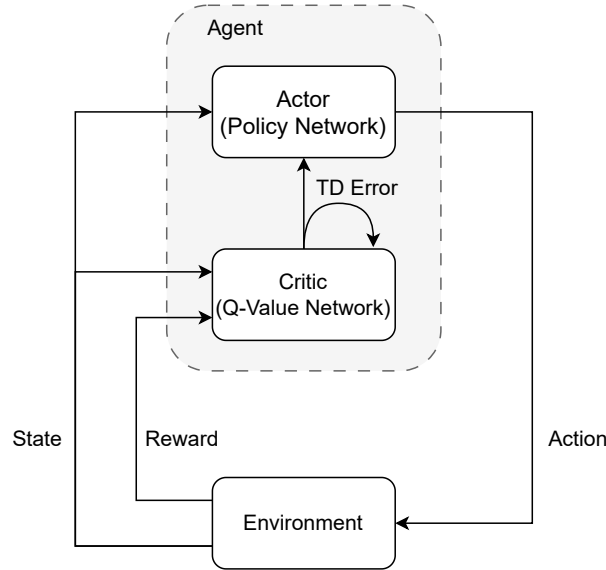


Figure 3.1: Actor-Critic Framework

Critic Update. SAC uses two soft Q-functions $Q_{\theta_1}(s, a)$ and $Q_{\theta_2}(s, a)$, updated to minimize the soft Bellman error using the target:

$$y(r, s', a') = r + \gamma \left(\min_{i=1,2} Q_{\theta_i}(s', a') - \alpha \log \pi_\phi(a'|s') \right) \quad (3.2)$$

The loss function for each Q-function is:

$$J_Q(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim D} \left[(Q_{\theta_i}(s,a) - y(r,s',a'))^2 \right] \quad (3.3)$$

where D is the replay buffer and θ'_i are the target Q-function parameters.

Target Q-Network Update. To stabilize learning, SAC maintains a set of target Q-networks with parameters θ'_1 and θ'_2 corresponding to the critic networks Q_{θ_1} and Q_{θ_2} . These target networks are updated using a soft update mechanism after each training step. The update is defined as:

$$\theta'_i \leftarrow \tau \theta_i + (1 - \tau) \theta'_i, \quad i = 1, 2 \quad (3.4)$$

where $\tau \in (0, 1]$ is the target smoothing coefficient that determines how quickly the target networks track the learned critics. A small value of τ (e.g., 0.005) ensures a slow and stable update, helping to prevent divergence during training.

Actor Update. The policy $\pi_\phi(a|s)$ is updated to maximize expected reward while maintaining entropy:

$$J_\pi(\phi) = \mathbb{E}_{s \sim D} [\mathbb{E}_{a \sim \pi_\phi} [\alpha \log \pi_\phi(a|s) - Q(s,a)]] \quad (3.5)$$

This encourages the policy to choose actions that yield high return while remaining stochastic.

Temperature Update. The temperature α can be automatically tuned by minimizing the following loss:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi} [-\alpha (\log \pi(a_t|s_t) + H_{\text{target}})] \quad (3.6)$$

where H_{target} is the desired target entropy, often chosen based on the action dimensionality.

SAC's combination of entropy maximization, function approximation, and stability enhancements like double Q-learning and target networks makes it highly suitable for continuous control problems in robotics.

3.1.3. Observation and Action Space

Observation space is the states of the environment that the agent perceives. In this thesis, the observation space is represented as a continuous space and consists

that since the peg and hole have cylindrical cross-section, there is no need for an angular velocity in the z -axis of the end-effector frame, as it will not affect the peg insertion. Furthermore, during training, the linear and angular velocity of the end-effector is limited to 0.1 m/s and 0.1 rad/s, respectively, to avoid excessive contact force and unreasonable motion.

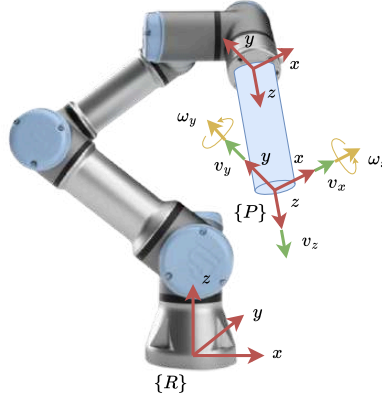


Figure 3.3: Schematic of Action Space

3.1.4. Reward Function

The reward function used in this thesis is defined as:

$$r(s, a) = \begin{cases} 100, & \text{if the task is successfully completed,} \\ -20, & \text{if a safety violation occurs,} \\ 1.5 r_{\text{distance}} + r_{\text{force}}, & \text{otherwise.} \end{cases} \quad (3.10)$$

The condition for task completion varies depending on the current stage of the curriculum, which is described in detail in the next section. A safety violation is defined as any instance where the torque in one or more robot joints exceeds its allowable limit.

The term r_{distance} represents a penalty based on the distance between the peg and the hole, computed as:

$$r_{\text{distance}} = - \left\| \frac{\Delta \mathbf{p}}{\mathbf{p}_{\text{max}}} \right\| \quad (3.11)$$

where $\Delta \mathbf{p} = [x, y, z]_h - [x, y, z]_p$ is the position error between the hole and the peg, and $\mathbf{p}_{\max}$ is a normalization vector that defines the maximum expected position offset in each axis:

$$\mathbf{p}_{\max} = [0.6, 0.6, 0.5]$$

The values for $\mathbf{p}_{\max}$ in the x and y axes are derived from the arena dimensions, while the value for the z -axis is based on the maximum vertical reach of the UR3e manipulator. This is based on the assumption that the hole's vertical position remains approximately aligned with the base of the robot.

Since r_{distance} is proportional to the (negative) normalized position error, the agent receives higher penalties when the peg is far from the hole, and lower penalties as it moves closer. This serves as a primary shaping reward to guide the agent toward the goal.

The term r_{force} penalizes excessive contact forces and torques at the end-effector. This term is designed to encourage compliant behavior during contact and reduce the risk of damage. Its exact definition also depends on the current curriculum stage and will be elaborated in the next section.

3.1.5. Domain Randomization

To train a policy that generalizes between different hole positions, this study applies domain randomization to the hole position. During training, the position of the hole is randomized within a predefined workspace at the beginning of every episode. The workspace is defined as an area surrounding the robot, with the boundaries expressed in polar coordinates centered on the robot's base frame. An illustration of how the workspace is defined can be seen in Figure 3.4. The workspace is thus defined by an area within $[r_{\min}, r_{\max}]$ and $[\theta_{\min}, \theta_{\max}]$.

Considering the capacity of the robot and the created environment (which will be detailed more in Section 3.2), the values were set to $r_{\min} = 0.3$ m, $r_{\max} = 0.4$ m, $\theta_{\min} = -\frac{5\pi}{6}$ rad, and $\theta_{\max} = -\frac{\pi}{6}$ rad. These values also reflect the available working area of the real robot situated in our laboratory.

Furthermore, the orientation of the hole is also randomized. This is done by adding a random offset in the range of $[-\frac{\pi}{20}, \frac{\pi}{20}]$ along its 3 axes (x, y, z). By doing so, the agent can learn a policy that is robust to uncertainties in the hole's orientation, especially when deployed into real systems.

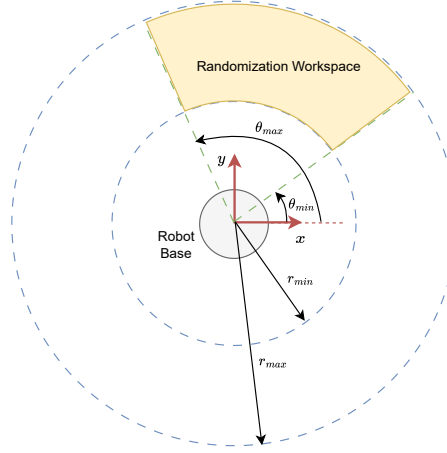


Figure 3.4: Randomization Workspace

3.1.6. Curriculum Implementation

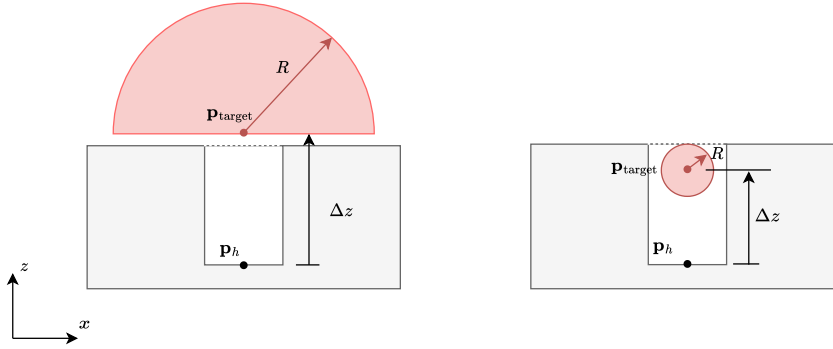
This thesis implements a curriculum to speed up learning. The curriculum consists of a total of eight stages with progressing difficulties. Each stage can be considered as a sub-task that serves as the foundation for the next stage.

Stages 1 to 7 aim to guide the agent to move the peg towards the hole with the appropriate trajectory. The task in each stage can be broken down into two parameters: target position $\mathbf{p}_{\text{target}}$ and range threshold R . Target position can also be expressed as the vertical offset Δz from the hole frame, which is located at the bottom of the hole. In each stage, the task is considered completed successfully if the peg tip reaches within the range threshold from the target position. An extra condition is added for the first two stages, in which the position of the peg tip along the global z -axis must be above the target position to prevent the end-effector from approaching the hole from below. When these conditions are achieved, the agent is rewarded with the maximum reward as listed in equation 3.10. The values of the task parameters in all stages can be seen in Table 3.1, and a schematic visualizing the task parameters can be seen in Figure 3.5.

Stage 8 builds on stage 7, with an addition of noise to the observed hole position. This noise is added as a position offset d along the xy -plane of the hole surface with a constant distance of 5 mm. This offset is given at a randomized angle θ at the beginning of every episode, and stays the same throughout the episode. A schematic of this scenario can be seen in Figure 3.6. Thus, instead of the actual hole position $\mathbf{p}_h$, the noisy hole position $\mathbf{p}_{h,\text{obs}}$ is added to the observation vector

Table 3.1: Task Parameters in the Curriculum (Up Until Stage 7)

Stage	Vertical Offset from Hole Δz (mm)	Range Threshold R (mm)
1	70	90
2	70	50
3	70	10
4	55	10
5	45	10
6	30	10
7	0	10

**Figure 3.5:** Schematic of Task Definition (left: stages 1-2, right: stages 3-7)

(i.e. as the first three elements of $\mathbf{x}_h^R$ in equation 3.7). The addition of noise to the observed hole position is expected to encourage the agent to perform a contact-rich hole search that is robust to position uncertainties, utilizing the observation of end-effector force and torque to obtain the precise location of the hole with respect to the peg.

Stage 8 also includes force penalization as mentioned in Section 3.1.4. The agent receives a small penalty r_{force} of -0.05 if the observed end-effector forces or torques exceed their safety thresholds. Mathematically, this can be expressed as

$$r_{\text{force}} = \begin{cases} -0.05, & \text{if } \exists i \in 1, \dots, 6, |\mathbf{f}_{\text{obs}}^{(i)}| \geq \mathbf{f}_{\text{lim}}^{(i)} \\ 0, & \text{otherwise} \end{cases} \quad (3.12)$$

where $\mathbf{f}_{\text{obs}}^{(i)}$ is the i -th element of the observed end-effector force/torque vector (where $\mathbf{f} = [F_x, F_y, F_z, M_x, M_y, M_z]$), and $\mathbf{f}_{\text{lim}}^{(i)}$ is its corresponding safety limit. Based

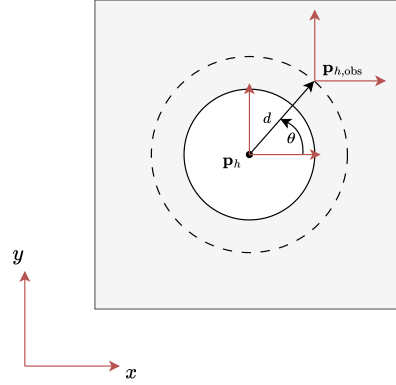


Figure 3.6: Schematic of Observation Noise

on the range of the force-torque sensor mounted in the manipulator’s tool flange, the safety limit is set to $\mathbf{f}_{\text{lim}}^{(i)} = [30, 30, 30, 10, 10, 10]$ during training. This force penalization would encourage the agent to generate actions that avoid high contact force, preventing any damage to the peg, hole, and the robot itself.

During training, the agent progresses sequentially from Stage 1 to Stage 8. At each stage, performance is evaluated at the end of every episode to determine whether the agent is ready to advance in the curriculum. Advancement is based on the success rate over the most recent n episodes. Specifically, the agent proceeds to the next stage if it achieves a success rate greater than 95 percent over the last 50 episodes, provided that it has trained for at least 50 episodes in the current stage.

3.2. Simulation Setup

In this thesis, the task is first simulated using MuJoCo simulator. Developed by Google DeepMind, MuJoCo (stands for Multi-Joint dynamics with Contact) [31] is a high-performance physics simulator designed for simulating articulated structures with complex contacts, such as robotic manipulators and biomechanical models. It is highly optimized for speed and accuracy in simulating complex, contact-rich interactions, making it well-suited for robotic control tasks. Its integration with common DRL libraries, such as OpenAI Gym and Stable-Baselines3, further supports rapid prototyping and benchmarking. Additionally, MuJoCo supports customization of environments and is widely used in academic benchmarks,

contributing to its popularity in the learning-based robotics community.

To create a simulated environment in MuJoCo, this thesis builds upon the template provided by the open-source *Manipulator-Mujoco* repository [32]. This repository offers a Gymnasium-based environment structure that can be adapted for reinforcement learning tasks. It is built on top of the *dm_control* library [33], which is a Python-based interface and environment suite developed by Google DeepMind for research in continuous control and reinforcement learning. The *Manipulator-Mujoco* repository also includes helpful tools such as a generic operational space controller and transformation utilities, which were used to support the development of the custom peg-in-hole environment implemented in this thesis.

The implemented custom environment follows the format of a Gymnasium environment class [34], which contains the following components:

- **Initialization**

The environment is initialized by setting up the task, the manipulator arm, the observation space, and the action space, along with some other variables and tools related to the learning algorithm, such as the domain randomizer and learning curriculum. This function also contains rendering options in order to visualize the environment in MuJoCo simulator.

- **Reset**

This method is called at the beginning of every episode to reset the environment. This includes setting the manipulator back to its initial configuration (set to $\mathbf{q}_{\text{initial}} = [-\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}, 0]$) and randomizing the position and orientation of the hole. Once the environment is reset, this method returns the initial observation.

- **Step**

This method is called to perform a single time step iteration inside the environment. A step consists of performing an action, computing the state of the environment after applying that action, computing the reward at that step, and evaluating termination criteria. This method returns a 5-tuple consists of observation, reward, terminated, truncated, and info.

- **Render**

This method contains the functionalities to visualize the states of the environment with the MuJoCo simulator. It is essential for visualizing the training process and the performance of the trained policy.

The full simulation setup can be seen in Figure 3.7. It is worth noting that since MuJoCo is unable to simulate collision of concave objects, the hole has to be decomposed into multiple convex parts. This decomposition was done using an open-source tool *obj2mjcf* [35]. However, due to the convex decomposition process, the original cylindrical hole is approximated using a polygonal shape — specifically, an octagonal hole in this case. The polygon’s outer boundary fits the original cylindrical hole’s circular profile. As a result, the actual inner clearance of the simulated hole becomes slightly smaller than the original circular hole. This problem is visualized in Figure 3.8, which shows the top view of the hole. The orange circle indicates the original circular hole, while the white dashed circle indicates the inner clearance due to convex decomposition. In this simulation, the original radius of the circular hole is 27 mm. Decomposing this circular hole into an octagonal hole creates an inner clearance of approximately 25 mm, as described in Section 3.1.1.

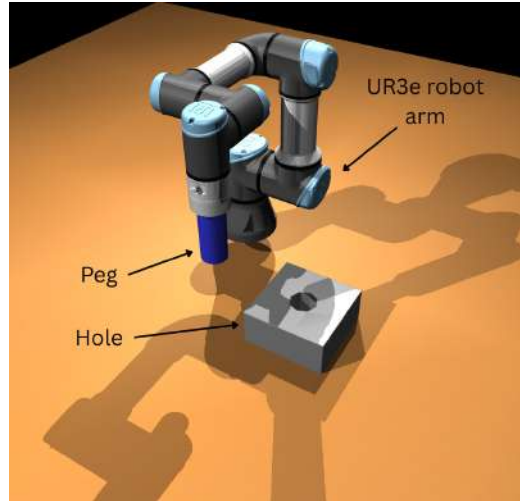


Figure 3.7: Simulation Environment Setup

Since training is done in the simulation, the SAC algorithm used to train the agent is implemented using the Stable-Baselines3 (SB3) library [36]. This library contains implementations of reinforcement learning algorithms in PyTorch, and along with it are the functionalities to train a policy and load a trained policy to be tested. The hyperparameter values used in this thesis follow the default values from SB3 implementation listed in Table 3.2. Additionally, during training and testing in simulation, each episode is limited to 250 timesteps, with each timestep corresponding to 0.025 seconds.

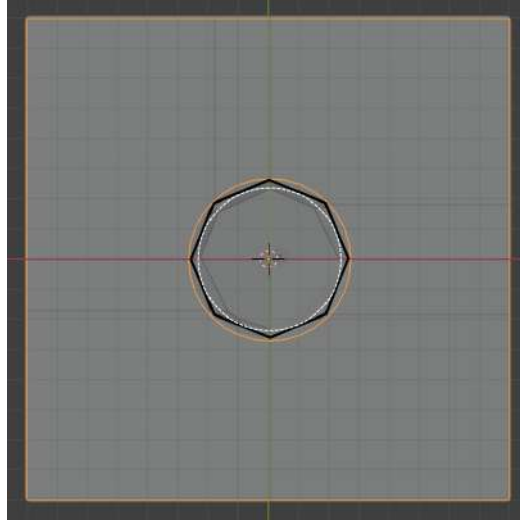


Figure 3.8: Result of Hole Decomposition Using *obj2mjcf* with a Smaller Inner Clearance

Table 3.2: SAC Hyperparameters Used in this Thesis

Hyperparameter	Value
learning rate	3×10^{-4}
replay buffer size	10^6
target smoothing coefficient (τ)	0.005
discount factor (γ)	0.99
actor and critic network architecture	[256,256]

3.3. Simulation-to-Real Transfer

3.3.1. Real-World Experiment Setup

To investigate the sim-to-real transferability of the policy trained in simulation, experiments were conducted with a real robot system. Figure 3.9 shows the setup used for the real-world experiments. The setup is designed to resemble the simulation environment as closely as possible. The following aspects were considered:

- **Workpieces:** The peg and hole box are fabricated using 3D-printed components, designed to match the exact dimensions used in the simulation (refer to Section 3.1.1).
- **Robot Joint Configuration:** The robot’s default joint configuration is set to align with the simulation setup (refer to Section 3.2). However, a

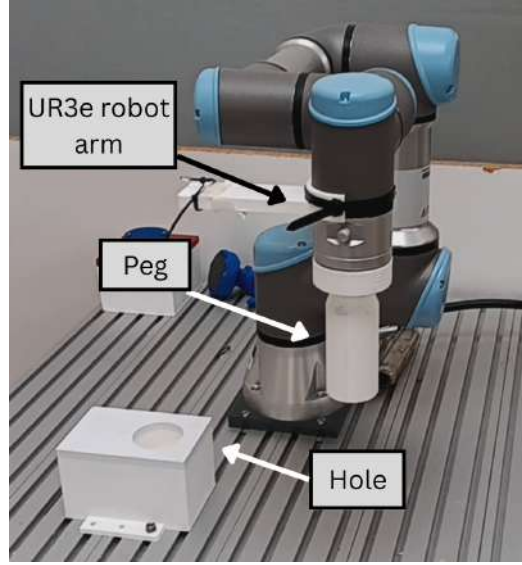


Figure 3.9: Real-world experimental setup.

discrepancy is observed in the joint state readings: the first joint angle (q_1) registers as $\frac{\pi}{2}$ on the physical robot but $-\frac{\pi}{2}$ in simulation. To ensure consistency with the training distribution, π is subtracted from the received angle before it is passed to the model.

- **End-Effector Position:** The UR3e ROS driver provides the end-effector pose via the TF tree. Because the physical end-effector (peg) extends beyond the robot flange, an additional static transform is defined to account for this offset. A TF listener retrieves the final pose of the peg, which is then incorporated into the model input.

3.3.2. Software Architecture

To connect and control the UR3e robot, the official ROS-based driver provided by Universal Robots [37] is utilized. The installation steps outlined in the repository are followed, including setting up the URCap software on the robot and installing the ROS driver package on the server computer.

Establishing the connection between the robot and the PC is straightforward: the two devices are connected via an Ethernet cable, and the PC’s network settings are configured to match the robot’s IP subnet. The ROS driver includes a launch file that initializes the controller and sets up communication with the robot. The *twist_controller* provided by the UR driver package is used to send velocity commands, and the launch file parameters are modified to load this controller by

default.

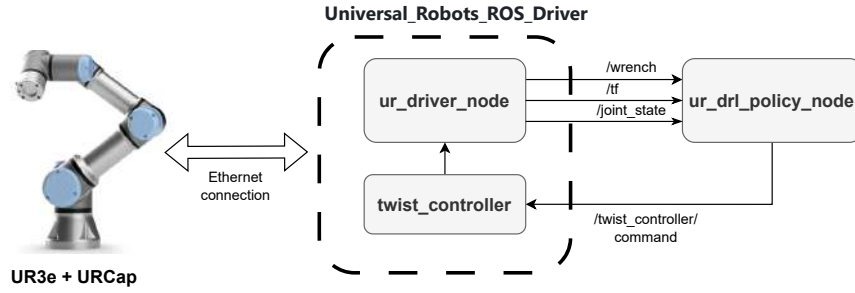


Figure 3.10: An overview of the ROS based software architecture used to interface with the UR3e robot.

To deploy the trained policy, a custom ROS node named *ur_drl_policy_node* is developed. This node receives robot state information, such as joint positions, end-effector pose, and end-effector wrench, from the driver node. These observations are used as input to the model, which generates action at a fixed control interval of 0.1 seconds. However, the action space of the trained policy is defined in the end-effector frame, whereas the *twist_controller* in the UR3e driver package requires velocity commands in the robot’s base frame. To address this mismatch, the generated velocity is transformed into the robot’s base frame by left-multiplying it with the rotation matrix representing the orientation of the peg relative to the robot base before being published to the controller.

Moreover, *ur_drl_policy_node* also generates two new services: *set_hole_pose* and *activate_policy*. At the beginning of every experiment, the hole pose must be set. This is done by moving the end-effector directly from the UR3e console (either in freedrive mode or in teleoperation mode) such that the tip of the peg touches the bottom of the hole and calling the *set_hole_pose* service. Once this is done, the robot should be returned back to the initial configuration, and *activate_policy* service can be called to start moving the robot with the generated action from the trained policy. This service can also be called to stop receiving robot control from the policy.

An overview of the software architecture used to interface with the UR3e robot is shown in Figure 3.10.

4. Results and Discussions

4.1. Simulation Results

During training, several metrics were logged and plotted in real time with Tensorboard to evaluate the training process. Mainly, there are three metrics to be tracked:

- **Average episode length:** the average number of timesteps in one episode. Normally, the learning algorithm aims to minimize this value.
- **Average episode reward:** the average total reward in one episode. Normally, the learning algorithm aims to maximize this value.
- **Success rate:** the ratio of successful episodes (i.e., an episode that ends with the task completed successfully, depending on the curriculum).

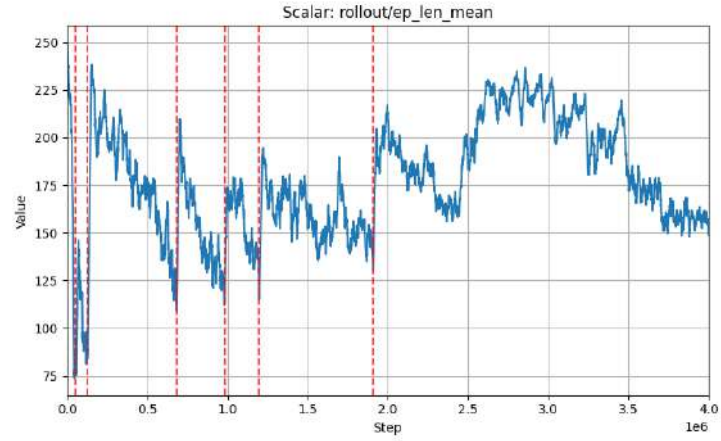
Per the default settings of the SB3 library, all the metrics above are computed and logged every 4 episodes.

After training, the policy was tested in simulation across five seeds, with 100 episodes in each seed. The performance of the trained policy was measured using the same performance metrics as listed above.

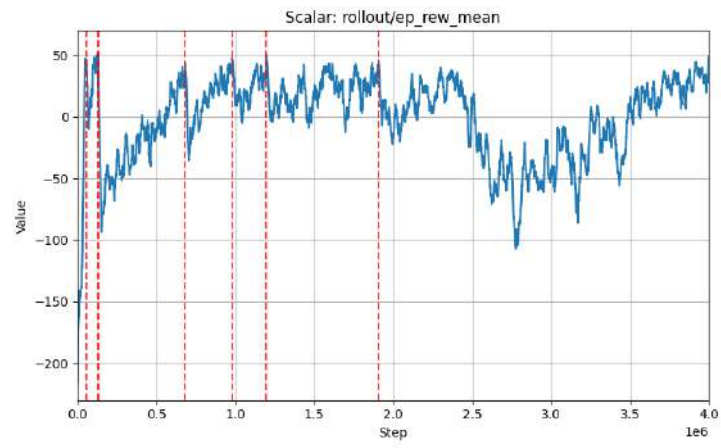
4.1.1. Peg-in-Hole Task Evaluation: Stages 1 to 7

As mentioned in Section 3.1, training of the earlier stages in the curriculum up until Stage 7 was done until the success rate in the last 50 episodes reached 95 percent. The plot of the training metrics up until Stage 7 can be seen in Figure 4.1. The red dashed lines indicate the timestep at which learning advanced a stage in the curriculum (as described in Section 3.1.6). As shown in the figure, training takes up to around 4 million timesteps, corresponding to approximately 6 hours of training performed on NVIDIA GeForce RTX 5080 (16 GB VRAM) GPU.

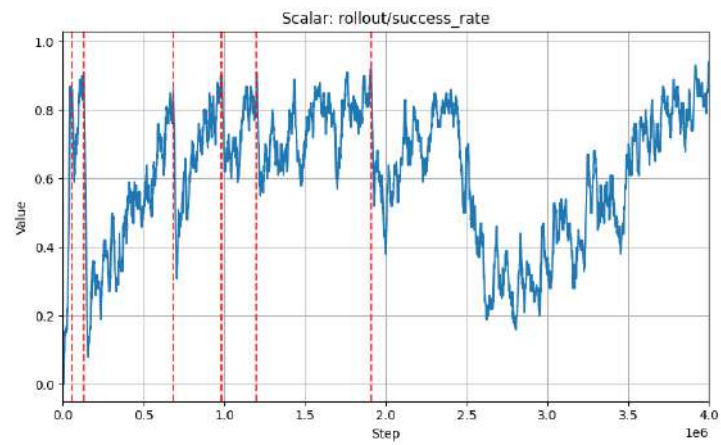
Some observations can be drawn from the plots. First, the beginning of each stage is often marked by a spike in average episode length, accompanied by a drop



(a) Average Episode Length



(b) Average Episode Reward



(c) Success Rate

Figure 4.1: Plot of Training Metrics Up Until Stage 7

in both average episode reward and success rate. This behavior is expected, as progression to a new stage introduces changes in task conditions. Specifically, up until this stage, states that give maximum reward in the previous stage no longer give maximum reward. As a result, the agent must re-explore the environment to discover new state-action pairs that optimize the revised reward function.

Secondly, it is evident that the number of timesteps required to satisfy the advancement criteria varies substantially across stages. Some stages (e.g., stages 1 and 2) are completed rapidly, suggesting they may be too easy or not sufficiently different from previous stages. Others require substantially more time, reflecting higher complexity or abrupt transitions in task difficulty. This uneven distribution of learning difficulty across stages suggests opportunities for refinement in the curriculum design. In particular, stages that take up much shorter durations may indicate that they were not challenging enough or not sufficiently distinct from preceding stages. These could be candidates for merging to streamline the learning process. Conversely, stages that exhibited slower learning (indicated by long plateau regions) may benefit from being further decomposed into smaller, more incremental sub-stages to provide a smoother learning progression.

Table 4.1: Model Stage 7 Testing Results (with Task Defined on Final Stage)

Seed	Success rate	Average episode length (steps)	Average episode reward
50	0.87	145.02	43.437
45	0.76	160.67	25.929
42	0.80	160.16	31.569
40	0.82	151.58	33.912
30	0.79	160.86	27.200
Mean	0.808	155.658	32.410
Std	0.041	7.111	6.959

Table 4.1 shows the performance metrics of the policy at this stage, tested with the task definition and reward structure of the final stage (i.e., with observation noise and force penalization). The policy achieves an average success rate of 0.808 across five seeds, with an average episode length of 155.658 timesteps (corresponding to 3.9 seconds of execution time) and average episode reward of 32.41. The relatively high success rate indicates that up to this training stage, the

agent is able to generalize between various hole poses to solve the peg-in-hole task. Breaking down the whole peg-in-hole task into different stages helps guide the agent to move in an appropriate trajectory in approaching the hole and inserting the peg fully into the hole. The resulting trajectory for one of the experiments conducted in the simulation can be seen in Figure 4.2.

4.1.2. Evaluation of Final Stage

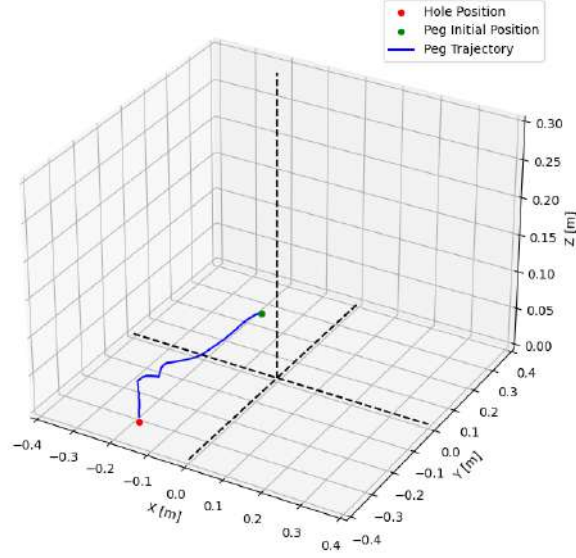
Training is extended to a final stage, with added observation noise and force penalization. This stage is intended to guide the agent toward learning contact-based search behaviors, particularly under observation noise, with minimum contact force to ensure compliant behavior. Training this stage was done for around 4 million timesteps, totalling 8 million timesteps between all stages. The resulting training metrics can be seen in Figure 4.3.

Ideally, the training metrics would converge to a certain value, indicating stable learning. However, the metrics during this stage do not exhibit clear signs of convergence. While the episode length shows moderate stabilization, both the success rate and reward remain noticeably volatile, indicating inconsistency in the learned policy. This suggests that despite the extended training duration, the introduction of observation noise and force penalization simultaneously may create difficulties that are not sufficiently supported by prior stages, causing the agent to struggle to generalize across different episodes.

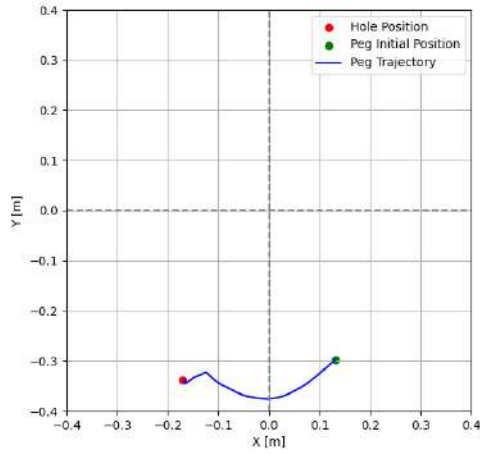
Table 4.2 shows the performance metrics of the final policy, tested with the task definition and reward structure of the final stage. Compared to stage 7, it is evident that this policy performs rather worse, indicated by significantly lower success rate and average episode reward.

To assess whether the agent acquired the force-compliant behavior, in addition to the performance metrics in Table 4.2, we also evaluate the average maximum force and torque magnitudes for each tested seed. These metrics are reported for both the policies from Stages 7 and final stage, which can be seen in Tables 4.3–4.4.

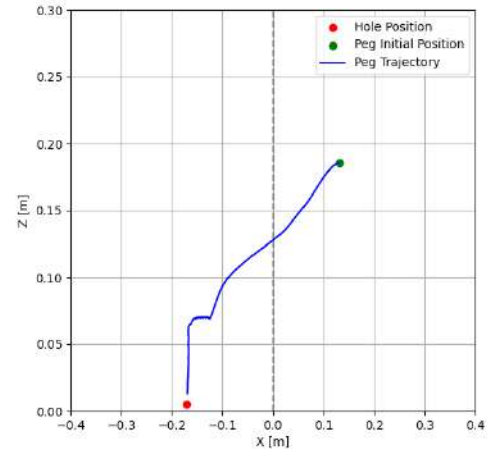
Comparing Table 4.4 to Table 4.3, there is a decrease of around 8-12 percent in the average maximum contact force, suggesting that the force penalization may have achieved its intended outcome of minimizing contact force. However, this decrease in contact force is accompanied with a significant reduce of success rate from 0.808 in Stage 7 (Table 4.1) to 0.65 in the final stage (Table 4.2). Simulation playback suggests that in the final stage, the agent occasionally avoids contact



(a) 3D View



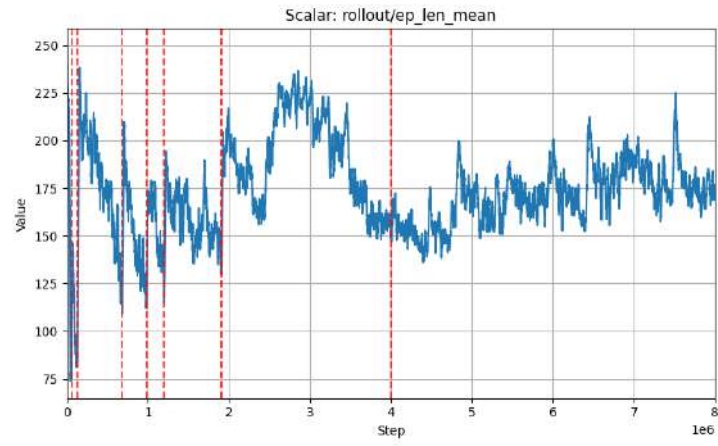
(b) XY Projection



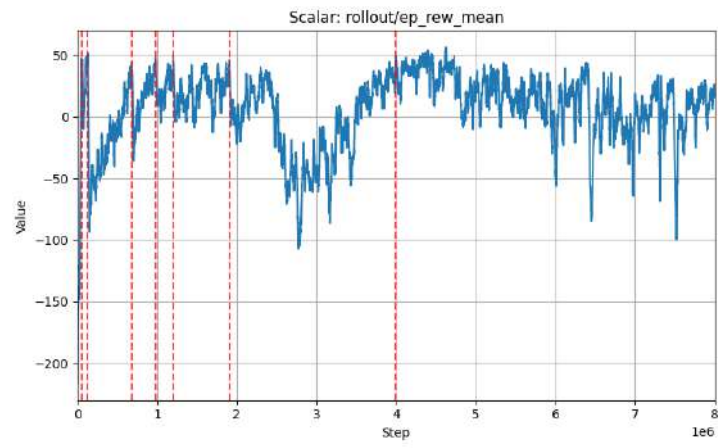
(c) XZ Projection

Figure 4.2: Peg Trajectory with Stage 7 Policy

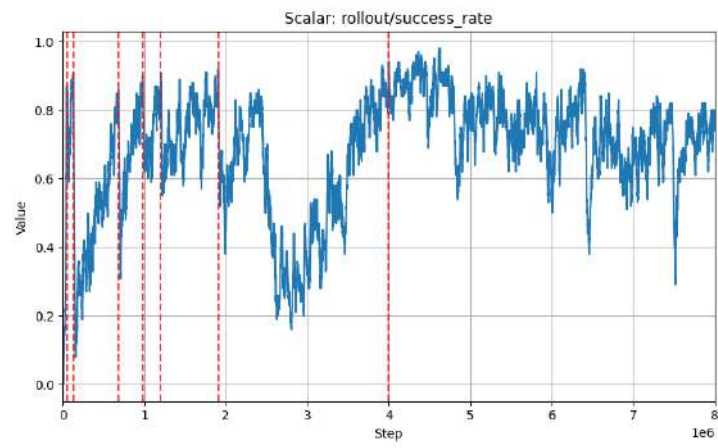
altogether, which likely contributes to the reduced success rate. These observations suggest that the final stage was only partially successful in encouraging more compliant search behavior.



(a) Average Episode Length



(b) Average Episode Reward



(c) Success Rate

Figure 4.3: Plot of Training Metrics Up Until Final Stage

Table 4.2: Final Policy Testing Results

Seed	Success rate	Average episode length (steps)	Average episode reward
50	0.67	171.3	10.744
45	0.62	178.82	1.681
42	0.60	178.29	3.883
40	0.71	172.04	13.550
30	0.65	180.72	6.125
Mean	0.65	176.234	7.197
Std	0.043	4.271	4.886

Table 4.3: Average Maximum Force-Torque for Stage 7 Policy

Seed	Average max force (magnitude) [N]			Average max torque (magnitude) [N.m]		
	F_x	F_y	F_z	M_x	M_y	M_z
50	26.930	26.723	36.906	2.478	2.456	0.379
45	24.999	25.460	32.531	2.358	2.294	0.348
42	25.624	23.477	31.567	2.145	2.275	0.327
40	25.515	24.646	34.204	2.321	2.373	0.359
30	25.806	24.112	34.252	2.251	2.362	0.355
Mean	25.775	24.884	33.892	2.311	2.352	0.353
Std	0.712	1.260	2.035	0.124	0.072	0.019

4.2. Test Results with Real Robot

To evaluate the transferability of the learned policy to the real world, we deployed the trained policy on a physical UR3e robot. The evaluation was conducted using 10 different hole positions within the trained spatial range (i.e., the randomization workspace as introduced in Section 3.1.5), introduced to assess the policy’s robustness to varying task conditions. Since the policy from Stage 7 gives the best success rate, this policy is tested on the real robot along with the policy from the final stage.

Table 4.4: Average Maximum Force-Torque for Final Stage Policy

Seed	Average max force (magnitude) [N]			Average max torque (magnitude) [N.m]		
	F_x	F_y	F_z	M_x	M_y	M_z
50	22.401	21.100	32.322	1.951	2.243	0.327
45	21.918	20.492	29.371	1.813	2.247	0.322
42	23.576	21.603	32.897	1.992	2.442	0.349
40	22.898	21.923	30.954	2.002	2.352	0.325
30	23.407	22.451	31.874	2.097	2.366	0.343
Mean	22.840	21.514	31.484	1.971	2.330	0.333
Std	0.691	0.753	1.378	0.103	0.085	0.012

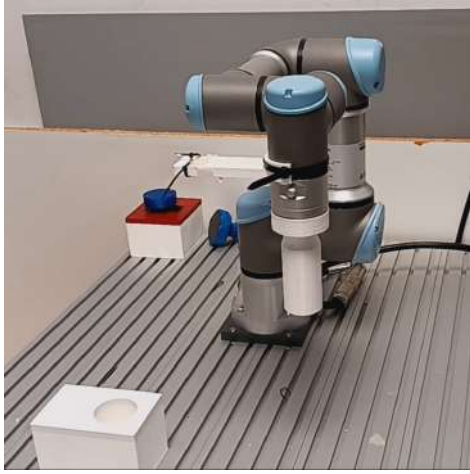
4.2.1. Evaluation of Transferred Policy

From the 10 experiments conducted, the policy from Stage 7 achieved success in 3 experiments. A sequence from one of the successful experiments can be seen in Figure 4.4, with the resulting trajectory shown in Figure 4.5. When the same hole pose is replicated in the simulation, the resulting trajectory can also be seen in Figure 4.5. This indicates a potential for deployability of a policy trained in simulation. Despite some successful trials, the significant drop in success rate between simulation (around 0.8; refer to Table 4.1) and real environment (around 0.3) indicates that there is still a huge discrepancy between simulation and real-world applications. Meanwhile, the policy from the final stage did not manage to achieve any successful trials.

4.2.2. Simulation-to-Reality Gaps

During (unreported) preliminary tests, we found that several modifications were necessary to ensure safe and stable execution in the real environment:

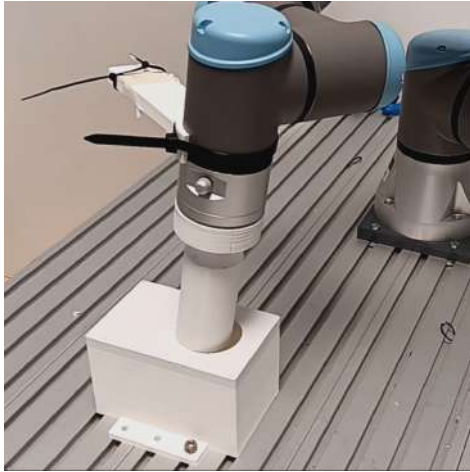
- The robot’s motion speed was scaled down to 10 percent (set from the UR console) of the commanded action. This was done to ensure safety of the robot and parts during motion.
- The frequency of action updates from the policy was reduced from 40 Hz in simulation to 10 Hz on the real robot. This adjustment helps accommodate system latency and prevents vibrations caused by unsmooth action



(a) Initial Position



(b) Peg Approaching the Hole



(c) Robot Performing Contact-Based Hole Search



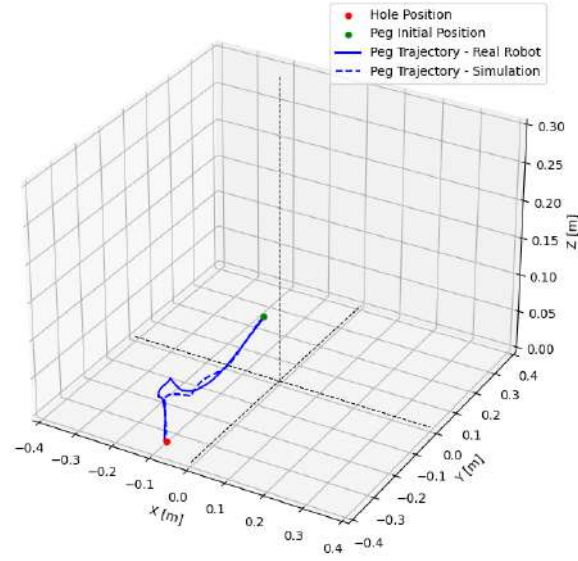
(d) Peg Successfully Inserted into the Hole

Figure 4.4: Real Robot Performing Peg-in-Hole Task Using Trained Policy

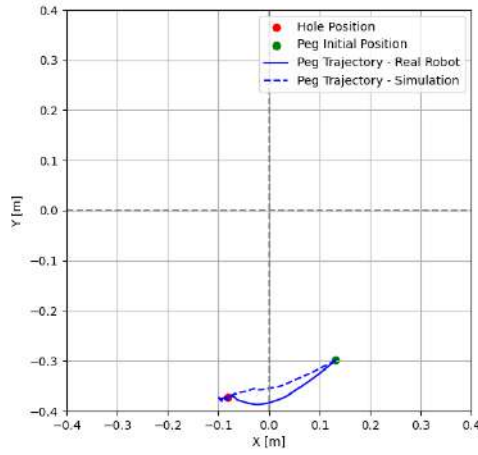
sequences.

- Limitation on episode length (i.e., number of timesteps in one episode) was not applied. Since the robot's speed was reduced, it was impossible to achieve the task within the number of episode steps as set in the simulation. Instead, for each hole position, the policy kept sending actions until the task was achieved, or until any safety violation occurred (usually indicated by a safety stop enforced by the built-in PolyScope software).

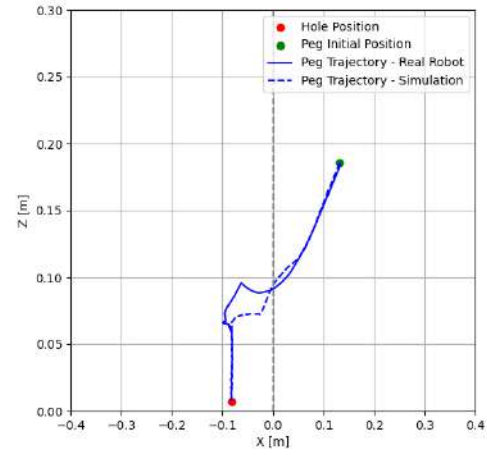
The most significant difference lies in the observation of end-effector force-torque. Figure 4.6a shows the evolution of end-effector force over time in one of



(a) 3D View



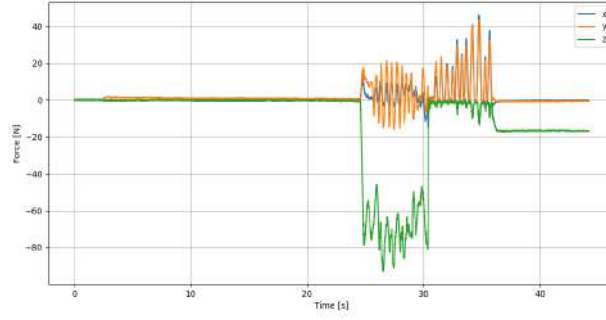
(b) XY Projection



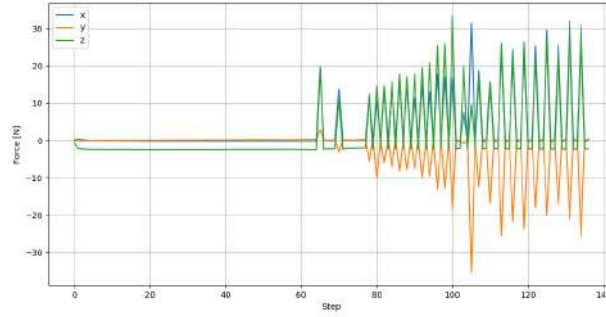
(c) XZ Projection

Figure 4.5: Peg Trajectory in Real World vs Simulation Generated with the Same Policy

the experiments that yielded success. This plot is generated using data straight from the ROS topic `/wrench`, which provides the force and torque at the tool flange expressed in the frame of the tool flange. When the same hole pose is replicated in simulation, the resulting end-effector force evolution can be seen in



(a) Real Robot



(b) Simulation

Figure 4.6: Plot of End-Effector Force in Real Robot vs Simulation

Figure 4.6b.

Two problems can be observed from this comparison. First, in the real robot, when contact takes place (at around 25 s in Figure 4.6a), the force in z-axis drops to a negative value. In contrast, in simulation, the same contact event results in a spike to a positive value (around step 62 in Figure 4.6b). A quick inspection of both environments suggests a sign mismatch in the force-torque sensor readings between the real and simulated systems. This mismatch can cause the policy to interpret contact incorrectly, potentially generating behaviors opposite to those intended. To fix this, the wrench data from the ROS topic are negated before being concatenated with other observations and passed to the policy.

Second, the magnitude of contact forces is significantly higher in the real robot. As seen in Figure 4.6, the maximum force magnitude in the real robot reaches around 90 N, while in the simulation, it only reaches around 35 N. This discrepancy suggests that the simulation does not accurately capture the physical dynamics of contact, particularly in terms of stiffness and compliance.

4.3. Insights from Training and Deployment Results

Based on the results found during training, it remains an open challenge to design the curriculum that strikes a balance between compliant behavior and task success. Future work should explore curriculum designs that better reinforce gentle contact without discouraging necessary interaction. A possible solution is to introduce observation noise and/or force penalization more gradually, either through intermediate stages or via adaptive reward shaping, to improve learning stability and generalization in the final task. Other alternative solutions may include adaptive thresholds, multi-objective rewards, or auxiliary feedback channels that capture the quality, not just the magnitude, of physical contact.

Additionally, training may benefit from hyperparameter optimization. SAC is known to be highly sensitive to hyperparameter choices [7]. Suboptimal configurations may lead to slower convergence, premature policy stagnation, or unstable behavior, which can compound across stages in a curriculum. Moreover, certain hyperparameters might interact differently with task complexities introduced at various stages, particularly with dense penalties like force penalization. While working on this thesis, several preliminary trainings that involve modifying the network architecture have been attempted, with insignificant results. However, future studies may involve tuning other hyperparameters as listed in Table 3.2. Systematic tuning, either through grid search, Bayesian optimization, or automated tools like Optuna [38] or Ray Tune [39], could help identify more effective parameter sets tailored to each stage or the curriculum as a whole. This could improve both sample efficiency and final policy robustness, especially in the more challenging late-stage tasks.

Furthermore, the various sources of sim-to-real gap identified above suggest two main directions for improvement. On the one hand, it may be necessary to refine the simulation environment to better match real-world conditions, for example, by improving contact modeling, sensor noise characteristics, and system latency. A more accurate simulation could enable the policy to learn behaviors that transfer more reliably to the physical robot. On the other hand, adapting or retraining the policy directly on the real system may still be required to fine-tune performance and account for unmodeled dynamics. Ultimately, a combination of both strategies may be needed to ensure robust real-world deployment.

5. Conclusion and Future Works

This thesis investigated the application of deep reinforcement learning (DRL) to solve the challenging peg-in-hole task, with a focus on achieving compliant behavior without the aid of conventional compliant controllers or demonstration-based initialization. The study proposed a curriculum learning approach to guide a single policy in learning multiple phases of the task, namely, the approach, search, and insertion, within a single unified framework.

The proposed approach showed that it is feasible to train a single DRL policy capable of executing both search and insertion phases. Extensive experiments in a physics-based simulation demonstrated the effectiveness of the proposed curriculum design, though several limitations were identified in learning stability and the handling of observation noise and force penalties.

Real-world experiments further revealed discrepancies between the simulated and physical environments, particularly in force dynamics and sensor interpretation. These sim-to-real gaps suggest that further refinement of both simulation fidelity and policy adaptation strategies is necessary for robust real-world deployment.

Overall, this work highlights both the promise and the challenges of using DRL for force-sensitive manipulation tasks, contributing insights that can inform future efforts in learning-based robot assembly. Future work should focus on refining curriculum design, exploring hyperparameter tuning strategies, and improving the fidelity of simulation to support more robust real-world performance.

BIBLIOGRAPHY

- [1] X. Liu, P. Huang, and Z. Liu, “A novel contact state estimation method for robot manipulation skill learning via environment dynamics and constraints modeling,” *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3903–3913, 2022.
- [2] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, “Deep reinforcement learning for high precision assembly tasks,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 819–825, 2017.
- [3] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz, “A practical approach to insertion with variable socket position using deep reinforcement learning,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 754–760, 2019.
- [4] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, “Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach,” *Applied Sciences*, vol. 10, no. 19, 2020.
- [5] A. Yuji Yasutomi, H. Mori, and T. Ogata, “A peg-in-hole task strategy for holes in concrete,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2205–2211, 2021.
- [6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, 02 2015.
- [7] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, “A survey on deep rein-

- forcement learning algorithms for robotic manipulation,” *Sensors*, vol. 23, no. 7, 2023.
- [8] H. Zhang, P. M. Kebria, S. Mohamed, S. Yu, and S. Nahavandi, “A review on robot manipulation methods in human-robot interactions,” 2023.
 - [9] X. Zhang, S. Jin, C. Wang, X. Zhu, and M. Tomizuka, “Learning insertion primitives with discrete-continuous hybrid action space for robotic assembly tasks,” in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9881–9887, 2022.
 - [10] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2019.
 - [11] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” 2018.
 - [12] A. Lämmle, P. Tenbrock, B. Bálint, F. Nägele, W. Kraus, J. Váncza, and M. F. Huber, “Simulation-based learning of the peg-in-hole process using robot-skills,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9340–9346, 2022.
 - [13] M. S. Alanoca Torrez, J. Yi, and X. Ji, “Integration of force-torque information into reinforcement learning for accurate position estimation using the td3 algorithm,” in *2024 4th Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS)*, pp. 313–318, 2024.
 - [14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1889–1897, PMLR, 07–09 Jul 2015.
 - [15] A. Franceschetti, E. Tosello, N. Castaman, and S. Ghidoni, “Robotic arm control and task training through deep reinforcement learning,” in *Intelligent Autonomous Systems 16* (M. H. Ang Jr, H. Asama, W. Lin, and S. Foong, eds.), (Cham), pp. 532–550, Springer International Publishing, 2022.
 - [16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017.

- [17] F. Wang, B. Cui, Y. Liu, and B. Ren, “Deep reinforcement learning for peg-in-hole assembly task via information utilization method,” *Journal of Intelligent & Robotic Systems*, vol. 106, p. 16, Aug 2022.
- [18] C. Chen, H. Zhang, Y. Pan, and D. Li, “Robot autonomous grasping and assembly skill learning based on deep reinforcement learning,” *The International Journal of Advanced Manufacturing Technology*, vol. 130, pp. 5233–5249, 01 2024.
- [19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, PMLR, 10–15 Jul 2018.
- [20] O. Petrovic, L. Schäper, S. Roggendorf, S. Storms, and C. Brecher, “Sim2real deep reinforcement learning of compliance-based robotic assembly operations,” in *2022 26th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 300–305, 2022.
- [21] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, “Learning force control for contact-rich manipulation tasks with rigid position-controlled robots,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, 2020.
- [22] F. Li, Q. Jiang, S. Zhang, M. Wei, and R. Song, “Robot skill acquisition in assembly process using deep reinforcement learning,” *Neurocomputing*, vol. 345, pp. 92–102, 2019. Deep Learning for Intelligent Sensing, Decision-Making and Control.
- [23] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, “Hindsight experience replay,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
- [24] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, “Data-efficient deep reinforcement learning for dexterous manipulation,” 2017.

- [25] A. Zhan, R. Zhao, L. Pinto, P. Abbeel, and M. Laskin, “Learning visual robotic control efficiently with contrastive pre-training and data augmentation,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4040–4047, 2022.
- [26] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, “Understanding domain randomization for sim-to-real transfer,” 2022.
- [27] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, “Curriculum learning for reinforcement learning domains: A framework and survey,” 2020.
- [28] P. Tu, Z. Sun, Y. Gao, P. Liu, R. Song, and Y. Song, “Robotic skill acquisition in peg-in-hole assembly tasks based on deep reinforcement learning,” *Procedia Computer Science*, vol. 250, pp. 129–135, 2024. Proceedings of the 2024 International Conference on Biomimetic Intelligence and Robotics.
- [29] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, “Accelerating robot learning of contact-rich manipulations: A curriculum learning study,” 2022.
- [30] C. Chen, H. Wang, Y. Pan, and D. Li, “Vision-based robotic peg-in-hole research: integrating object recognition, positioning, and reinforcement learning,” *The International Journal of Advanced Manufacturing Technology*, vol. 135, pp. 1119–1129, Nov 2024.
- [31] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.
- [32] “Manipulator-mujoco.” <https://github.com/ian-chuang/Manipulator-Mujoco.git>. Accessed: 2025-06-07.
- [33] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, “dm_control: Software and tasks for continuous control,” *Software Impacts*, vol. 6, p. 100022, 2020.
- [34] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, *et al.*, “Gymnasium: A standard interface for reinforcement learning environments,” *arXiv preprint arXiv:2407.17032*, 2024.

- [35] “obj2mjcf.” <https://github.com/kevinzakka/obj2mjcf.git>. Accessed: 2025-06-08.
- [36] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
- [37] U. Robots, “Universal robots ros driver.” https://github.com/UniversalRobots/Universal_Robots_ROS_Driver, 2020. Accessed: 2025-06-08.
- [38] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
- [39] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, “Tune: A research platform for distributed model selection and training,” *arXiv preprint arXiv:1807.05118*, 2018.

8

Sim2Real Transfer Learning for Object Tracking in Robotic Applications

Mir Mohibullah Sazid



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTER IN
INTELLIGENT FIELD ROBOTIC SYSTEMS (IFROS)

Sim2Real Transfer Learning for Object Tracking in Robotic Applications

Author:

Mir Mohibullah Sazid

BSc. in Electrical and Electronics Engineering.

Internal supervisor:

Dr. Zoltán Istenes

Eötvös Loránd University

External supervisor:

Dr. Dániel Horváth

HUN-REN SZTAKI

Budapest, 2025

Abstract

Real-world object tracking in robotics is heavily constrained by the scarcity of annotated datasets. Collecting and labeling real-world data is often expensive, time-consuming, and operationally impractical, which significantly hinders the development of robust learning-based tracking systems. To address this challenge, I propose a scalable sim2real transfer framework that leverages synthetic data generation powered by domain randomization. This approach enables the automatic creation of large-scale, richly annotated datasets within minutes, greatly reducing reliance on real-world supervision and narrowing the sim2real gap.

My methodology supports both detection-based and segmentation-based tracking models trained exclusively on synthetic data. In a zero-shot setting, the detection-based tracker achieves a standard MOTA of 0.799 and macro-MOTA of 0.685, while the segmentation-based tracker yields a standard MOTA of 0.801 but a lower macro-MOTA of 0.501. When fine-tuned using just a single annotated real-world frame (one-shot transfer), the detection-based tracker improves substantially, reaching 0.940 standard MOTA and 0.913 macro-MOTA, clearly outperforming its segmentation-based counterpart, which reaches 0.906 standard MOTA and 0.874 macro-MOTA. These findings highlight the superior generalization capabilities and data efficiency of detection-based tracking under minimal real-world supervision.

Extensive ablation studies further reveal the critical importance of photometric and spatial randomization techniques—such as lighting variation, object placement, and hue shifts—in enhancing model robustness and ensuring effective sim2real generalization. Although the work presented is entirely my own, plural pronouns are used throughout the thesis for stylistic consistency.

Acknowledgements

I would like to express my deepest gratitude to all those who have contributed to the successful completion of this thesis.

First and foremost, I am immensely grateful to my academic supervisor, **Dr. Zoltán Istenes**, whose unwavering support, insightful guidance, and expert advice have been instrumental in shaping this research. I would also like to extend my sincere thanks to my industrial supervisor, **Dr. Dániel Horváth**, for his invaluable mentorship and encouragement throughout this journey.

I am deeply thankful to the institutions that have provided a nurturing and stimulating environment for my work. I extend my appreciation to **HUN-REN SZTAKI**, the **University of Girona**, and **Eötvös Loránd University** for their exceptional resources and support. Additionally, I am grateful to **all the professors** of the **IFROS Masters program** for their constant guidance, constructive feedback, and dedication to advancing the field of robotics.

I would also like to acknowledge **my classmates**, whose friendship, collaboration, and insightful discussions enriched my academic experience and made this journey even more rewarding.

On a personal note, I wish to thank **my family** for their endless love and encouragement. To **my parents**, your unwavering support and belief in me have been a constant source of strength; and to **my sister**, your care and inspiration have made every step of this journey worthwhile.

This thesis is a testament to the collective efforts of so many wonderful individuals and institutions, and I am deeply grateful to each and every one of you.

CONTENTS

Contents

Abstract	i
Acknowledgements	
List of Abbreviations	v
List of Notations	viii
List of Figures	ix
List of Tables	xi
List of Algorithms	xii
 1 Introduction	 1
1.1 Background and Context	1
1.2 Problem Statement	2
1.3 Research Questions	2
1.4 Objectives and Scope	3
1.5 Contributions of the Thesis	4
1.6 Thesis Structure Overview	4
 2 Background Theory	 6
2.1 Transfer Learning	6
2.1.1 Formal Definitions and Notation	7
2.1.2 Homogeneous vs. Heterogeneous TL	7
2.1.3 Zero-shot and One-shot Transfer	7
2.2 Visual Perception Tasks in Computer Vision	8
2.3 Object Tracking	8
2.4 YOLOv11 Architecture for Backbone	9

CONTENTS

2.5	BotSORT for Multi-Object Tracking	11
2.6	Evaluation Metrics	11
3	Literature Review	12
3.1	Synthetic Data Generation	12
3.2	Empirical Benchmarks	13
3.3	Research Gaps and Summary	14
4	Methodology	17
4.1	Sim2Real Knowledge Transfer	17
4.2	Synthetic data generation	18
4.2.1	Domain Randomization Techniques	18
4.2.2	Camera Setup and Capture	23
4.3	Training	24
4.3.1	Training Protocol	25
4.4	Multiple Object Tracking	26
5	Dataset	27
5.1	Annotated Test Data	27
5.2	3D CAD Models for Synthetic Data Generation	28
6	Results and Evaluation	30
6.1	Dataset Variants and Randomization Composition	30
6.2	Evaluation Protocol	32
6.3	Detection-Based Tracking on Synthetic Domain	32
6.4	MOT configuration	33
6.5	Detection-Based Tracking Performance	35
6.5.1	One-Shot Transfer Evaluation	37
6.6	Segmentation-Based Tracking on Synthetic Domain	37
6.7	Segmentation-Based Tracking Performance	39
6.7.1	One-Shot Evaluation	40
6.8	Detection vs. Segmentation Tracking Comparison	41
6.9	Qualitative Evaluation	42
7	Application	44
7.1	System Overview	44

CONTENTS

7.2	Pick and Place Operations	44
7.3	Results	46
7.3.1	Joint-Position Tracking and End-Effector Trajectory	47
8	Conclusion and Future Work	49
8.1	Conclusion	49
8.2	Future Work	50
8.3	Closing Remarks	50
A	Evaluation Metrics	51
B	Synthetic Data Capture and Material Assignment Logic	53
	Bibliography	55

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AABB	Axis-Aligned Bounding Box
AP	Average Precision
AP50	Average Precision at $\text{IoU} \geq 0.5$
CAD	Computer-Aided Design
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CSP	Cross Stage Partial
DR	Domain Randomization
F1	F1 Score
FN	False Negative
FP	False Positive
FOV	Field of View
FPS	Frames Per Second
GAN	Generative Adversarial Network
GT	Ground Truth
HSVA	Hue, Saturation, Value, Alpha

List of Abbreviations

IDF1	Identification F1 Score
IDFN	Identity False Negative
IDFP	Identity False Positive
IDTP	Identity True Positive
IK	Inverse Kinematics
IoU	Intersection over Union
JDT	Joint Detection and Tracking
mAP	Mean Average Precision
MARL	Multi-Agent Reinforcement Learning
MOGPE	Multi-Object Grasp Pose Estimation
MOT	Multi-Object Tracking
MOTA	Multi-Object Tracking Accuracy
MOTP	Multi-Object Tracking Precision
NLP	Nonlinear Programming
OS	One-Shot
PNG	Portable Network Graphics
QP	Quadratic Programming
R-CNN	Region-based Convolutional Neural Network
ReID	Re-Identification
RL	Reinforcement Learning
RLE	Run-Length Encoding
RT	Real-Time
SBS-50	Strong Baseline Series 50

List of Abbreviations

STD	Standard Deviation
TL	Transfer Learning
TP	True Positive
UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once
ZS	Zero-Shot

List of Notations

Symbol	Description
X	Feature space
$P(X)$	Marginal probability distribution over features
Y	Label space
$P(Y X)$	Conditional probability of labels given features
$f(\cdot)$	Predictive function mapping input features to labels
$D = \{X, P(X)\}$	A domain composed of feature space and its distribution
$T = \{Y, P(Y X)\}$	A task composed of label space and conditional distribution
D_S, D_T	Source and target domains
T_S, T_T	Source and target tasks
$f_T(\cdot)$	Target predictive function to be optimized
(x_i, y_i)	Data sample: input feature and label pair
α, β, γ	Euler angles (roll, pitch, yaw)
q	Quaternion representing 3D object orientation
(x, y, z)	3D position of an object in Cartesian space
$S = \{(x_i, y_i)\}$	Set of Poisson Disk sample points
d	Minimum separation distance in Poisson sampling
w, h	Width and height of the object placement area
$\theta^*(t)$	Desired joint angle configuration at time t
$U(a, b)$	Uniform distribution in the range $[a, b]$
K	Camera Intrinsic Matrix
$V1, V2$	Test video sequences 1 and 2
Δ	Change or improvement indicator (e.g., ΔMOTA)

List of Figures

2.1	Progression of visual perception tasks: classification, localization, object detection, and instance segmentation [18].	8
2.2	Overall architecture of YOLOv11 [23].	10
2.3	Detailed structure of the C3K2 block [23].	10
2.4	BotSORT pipeline: detections undergo IoU matching, ReID re-matching, egomotion projection, and adaptive tracklet management [20].	11
4.1	Overview of sim2real knowledge transfer. Adopted to MOT from [38]	18
4.2	Comparison of placing objects	23
5.1	Challenging visual conditions present in the test dataset	28
5.2	Class distribution across the combined test dataset.	29
5.3	Visual comparison between simulation-based synthetic objects (top) and their corresponding real-world counterparts (bottom). All individual object images were resized to (256×256 pixels) for consistent visual alignment.	29
6.1	Examples of synthetic data generated across different datasets.	32
6.2	Training and validation loss, and evaluation metric trends over epochs.	33
6.3	Model training evaluation	34
6.4	ZS tracking performance on test sequences.	37
6.5	ZS vs. OS overall tracking metrics.	37
6.6	Segmentation model evaluation curves.	38
6.7	ZS overall tracking metrics	40
6.8	ZS vs OS segmentation tracking metrics on test sequences	41
6.9	Segmentation-based vs. detection-based tracking.	41
6.10	Qualitative tracking results under bounding box and segmentation-based configurations.	43

LIST OF FIGURES

7.1	Visual trace of the simulated pick-&-place sequence.	46
7.2	Joint-position tracking during the pick-and-place sequence. Solid lines denote <i>actual</i> joint angles, dashed lines the <i>IK-desired</i> angles, and dotted red lines indicate the joint limits (± 2.97 rad).	47
7.3	Three-dimensional end-effector (EE) path (blue) with object <i>pick</i> locations (red circles) and <i>place</i> targets (green triangles). The view is aligned with the world frame used in simulation.	47

List of Tables

3.1	Key Studies in Sim2Real	15
4.1	Data Augmentation Techniques for Training YOLOv11	25
5.1	Summary of Annotated Test Videos	28
6.1	Domain Randomization Parameters Used Across Datasets	31
6.2	Randomization Techniques Across Synthetic Datasets	31
6.3	Training Summary for Object Detection	32
6.4	BotSORT Tracker Configuration	34
6.5	Tracking Performance Summary Across Configurations	35
6.6	Class-wise MOTA Scores Across D2 and D3	36
6.7	Class-wise MOTA Improvement (%) from ZS to OS	38
6.8	Training Summary for Instance Segmentation	38
6.9	Segmentation-Based Tracking Performance on V2 (Zero-Shot)	39
6.10	Class-wise MOTA Improvement (%) from ZS to OS	41
B.1	Assigned Material Colors for 3D CAD Models	53

List of Algorithms

1	Object Placement via Poisson Disk Sampling	19
2	Rotation Randomizer	20
3	Hue Offset Randomizer	20
4	Color Randomizer	21
5	Lighting Randomizer	21
6	Background Texture Randomizer	22
7	Object Overlapping Placement	23
8	Compute Axis-Aligned Bounding Box (AABB)	24
9	Pick-and-Place with YOLO + BotSORT Tracking	45
10	Image Capture Core Logic	54

Chapter 1

Introduction

This chapter introduces the motivation, scope, and significance of robust object tracking within robotic applications. It outlines the challenges posed by the limited availability of real-world annotated data and describes the sim2real transfer learning framework proposed to address these challenges. Furthermore, this chapter presents the specific research questions, objectives, and key contributions of the thesis, concluding with an overview of the thesis structure.

1.1 Background and Context

Robust object tracking is an essential capability in modern robotics, supporting critical applications such as autonomous navigation, manipulation, surveillance, and human-robot collaboration. Reliable tracking systems enable robots to consistently identify and follow objects through various visual conditions and movements [1]. Achieving this capability typically requires extensive, high-quality datasets to train robust deep learning models [2, 3].

Synthetic data generation has emerged as an effective approach to creating extensive and diverse datasets, significantly reducing the reliance on manually collected and annotated real-world data [4, 5]. By using virtual environments, researchers can rapidly produce large quantities of accurately annotated images, capturing diverse visual scenarios. A central technique used in synthetic data generation is domain randomization, which systematically varies simulation parameters such as object appearance, lighting conditions, textures, and camera perspectives [6, 7]. This en-

1. Introduction

courages models to learn generalized, domain-invariant features, enhancing their robustness and versatility when deployed across different real-world environments.

The widespread use of synthetic datasets and domain randomization strategies represents a significant advancement in robotic perception and machine learning, providing scalable solutions that can be effectively adapted to various real-world applications [8].

1.2 Problem Statement

The central problem addressed in this research is the challenge of transferring object tracking models trained in synthetic environments to real-world robotic applications. Although synthetic data, coupled with domain randomization, enables the efficient generation of large-scale training datasets, models trained under these conditions often experience performance degradation when exposed to the visual complexities of the real world [7, 8]. This performance drop is primarily attributed to the reality gap—a statistical discrepancy between synthetic and real data distributions. Furthermore, there is limited understanding of which domain randomization strategies are most effective for enhancing sim2real transfer specifically in the context of object tracking [6].

Another dimension of this problem lies in evaluating the relative effectiveness of different tracking architectures. Detection-based methods like YOLO [9] and segmentation-based approaches such as Mask R-CNN [10] exhibit different trade-offs in terms of accuracy and computational efficiency, but their performance under sim2real conditions—especially with minimal real-world supervision—remains insufficiently explored. In addition, identity association methods commonly used in multi-object tracking pipelines [1] have not been thoroughly assessed when paired with detectors trained exclusively on synthetic data. These open challenges underscore the need for a deeper investigation into robust sim2real object tracking pipelines that are effective, scalable, and annotation-efficient.

1.3 Research Questions

1. **Sim2real Transfer:** How can domain-randomized synthetic datasets be systematically designed to maximise sim2real transfer performance for multi-

object tracking in robotic applications?

2. **Domain Randomization Effectiveness:** To what extent do domain-randomization strategies narrow the reality gap, and which factors—e.g., texture variation, illumination changes, sensor noise, or object kinematics—contribute most to improved real-world generalization?
3. **Architecture Trade-offs:** How do detection-based trackers compare with instance-segmentation-based trackers in terms of accuracy, computational efficiency, and data efficiency under sim2real transfer constraints?
4. **Identity Association Robustness:** How robust are state-of-the-art identity-association algorithms when paired with detectors trained exclusively on synthetic data, particularly in scenes characterised by occlusion, clutter, and camera motion?

1.4 Objectives and Scope

The primary objectives of this thesis are to:

- Build a scalable synthetic-data pipeline that automates the generation of richly annotated images, using extensive domain-randomization (illumination, pose, texture, colour and camera geometry).
- Advance sim2real transfer for robotic object tracking by systematically analysing how the synthetic-only models perform on physical video streams and by quantifying the residual reality gap.
- Compare two complementary tracking paradigms—bounding-box (detection-based) versus mask (instance-segmentation-based)—within a unified experimental framework.
- Create an annotated, publicly available benchmark dataset for video-level object tracking in robotic applications, evaluated using standard MOT metrics (MOTA, MOTP, IDF1).
- Demonstrate practical impact in an industrial scenario, validating the end-to-end pipeline on a pick-and-place case study.

The scope of this thesis encompasses the design, implementation, and experimental evaluation of the synthetic data generation pipeline and the associated tracking frameworks. Designed experimental validation is conducted within the context of robotic applications, where the proposed methods are rigorously tested in real-world scenarios to demonstrate their practical applicability.

1.5 Contributions of the Thesis

This thesis makes the following specific contributions:

- A synthetic-data generator for tracking that converts any CAD model into 7k labelled frames in minutes, freeing developers from manual annotation.
- The first exhaustive ablation of photometric and spatial randomizers for sim2real multi-object tracking, revealing the most influential parameters.
- A dual-pipeline comparison between detection-based tracking and segmentation-based tracking, both built on a unified YOLOv11 backbone integrated with BotSORT, enabling a fair and direct assessment of accuracy, speed, and data efficiency.
- A new publicly available annotated video dataset comprising over 2,500 frames and 23,000 annotations is introduced to support rigorous evaluation of object tracking algorithms using standard MOT metrics.

1.6 Thesis Structure Overview

This thesis is organised into eight chapters as follows:

- **Chapter 1 – Introduction:** Introduces the background, problem statement, objectives, research questions, and contributions of the thesis.
- **Chapter 2 – Background Theory:** Provides the theoretical foundation including transfer learning, object tracking, the YOLOv11 detection architecture, BotSORT tracking algorithm, and evaluation metrics.

1. Introduction

- **Chapter 3 – Literature Review:** Reviews existing work on synthetic data generation, domain randomisation, sim2real transfer, and multi-object tracking methods, highlighting open challenges.
- **Chapter 4 – Methodology:** Describes the proposed synthetic data generation pipeline, domain randomisation techniques, and camera setup used to create training datasets.
- **Chapter 5 – Dataset:** Introduces the real and synthetic datasets used for evaluation, including details of the annotated test data and 3D CAD models.
- **Chapter 6 – Results and Evaluation:** Presents experimental results on detection-based and segmentation-based tracking pipelines across synthetic and real domains, along with ablation studies and qualitative analysis.
- **Chapter 7 – Application:** Demonstrates the deployment of the trained models in a robotic pick-and-place scenario and evaluates their performance.
- **Chapter 8 – Conclusion and Future Work:** Summarises the key findings of the thesis and outlines potential future research directions.

Chapter 2

Background Theory

This chapter outlines the theoretical foundations of the proposed MOT system within a sim2real transfer learning context. It introduces transfer learning, the architecture of the YOLOv11 with emphasis on its modular structure and the efficient C3K2 block, the BotSORT tracking algorithm and it defines key evaluation metrics used to assess tracking performance in later experiments.

2.1 Transfer Learning

One of the fundamental challenges in robotics and machine learning is to enable a model to transfer knowledge acquired in one domain or task to another. This concept is known as *Transfer Learning* (TL). Rather than learning everything from scratch, TL allows leveraging previously acquired knowledge to improve learning efficiency and performance in new but related problems [11].

To illustrate the essence of TL, consider a student who is proficient in playing the guitar attempting to learn the piano, versus a student with no musical background. The former is likely to progress faster due to transferable knowledge such as rhythm and notation understanding [12].

In machine learning, TL is crucial when data in the target domain is scarce, expensive, or difficult to obtain. Unlike traditional supervised learning which assumes training and test data come from the same distribution, TL allows for a more relaxed assumption where the source and target distributions may differ [13]. It offers several benefits including faster training time, reduced need for large-scale labeled data in the target domain, lower computational cost, and improved ecological efficiency.

2. Background Theory

2.1.1 Formal Definitions and Notation

A *domain* $\mathcal{D}$ is defined as a pair $\mathcal{D} = \{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ denotes the feature space and $P(X)$ is the marginal probability distribution over the feature vectors $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}$.

A *task* $\mathcal{T}$ is defined as $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$, where $\mathcal{Y}$ is the label space, and $P(Y|X)$ is the conditional distribution that models the relationship between features and labels through a predictive function $f(\cdot)$.

Given a source domain $\mathcal{D}_S = \{\mathcal{X}_S, P(X_S)\}$ and task $\mathcal{T}_S = \{\mathcal{Y}_S, P(Y_S|X_S)\}$, and a target domain $\mathcal{D}_T = \{\mathcal{X}_T, P(X_T)\}$ and task $\mathcal{T}_T = \{\mathcal{Y}_T, P(Y_T|X_T)\}$, the goal of transfer learning is to improve the performance of a target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where either:

$$\mathcal{D}_S \neq \mathcal{D}_T \quad \text{or} \quad \mathcal{T}_S \neq \mathcal{T}_T$$

In cases where $\mathcal{T}_S = \mathcal{T}_T$ but $\mathcal{D}_S \neq \mathcal{D}_T$, the task remains the same while the domains differ. This is known as *domain transfer learning*.

2.1.2 Homogeneous vs. Heterogeneous TL

If $\mathcal{X}_S = \mathcal{X}_T$ but $P(X_S) \neq P(X_T)$, the TL problem is called **homogeneous**. However, if $\mathcal{X}_S \neq \mathcal{X}_T$ and/or $P(X_S) \neq P(X_T)$, it becomes **heterogeneous**, which often requires *feature space adaptation* [14]. These differences can result in:

- *Covariate shift*: $P(X_S) \neq P(X_T)$ while $P(Y|X)$ remains the same.
- *Conditional shift*: $P(Y|X_S) \neq P(Y|X_T)$.
- *Context bias*: Feature relevance differs between domains [15].

If the source and target domains or tasks are not sufficiently related, TL may result in *negative transfer*, potentially degrading the performance of the model [16].

2.1.3 Zero-shot and One-shot Transfer

Two important transfer paradigms are *zero-shot transfer* and *one-shot transfer*. In zero-shot transfer, the model has access to no labeled data from the target domain $\{\mathcal{D}_T, \mathcal{T}_T\}$ during training. In contrast, one-shot transfer allows the model to learn

from only one or a few examples from the target domain. These techniques are highly desirable in sim2real settings, where real-world labeled data is limited or expensive to collect [17].

2.2 Visual Perception Tasks in Computer Vision

Figure 2.1 illustrates the fundamental visual recognition tasks that form the foundation of many computer vision pipelines. Image classification involves identifying the primary object present in an image without specifying its position—for example, simply recognizing that an image contains a **CAT**. Classification with localization extends this by predicting both the object label and a bounding box that indicates where the object is located.

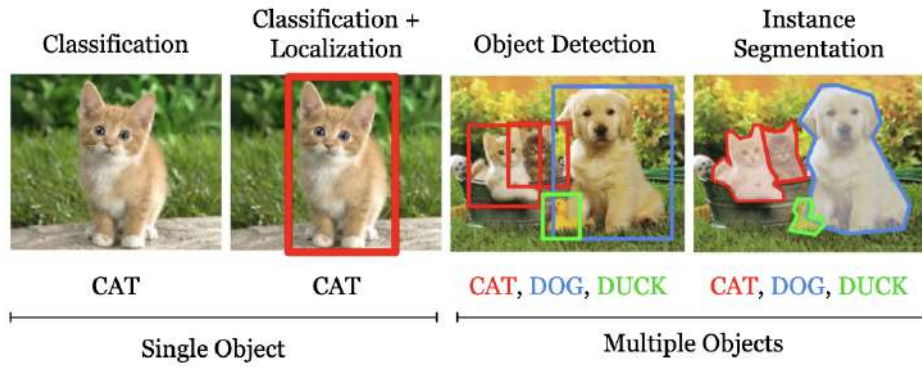


Figure 2.1: Progression of visual perception tasks: classification, localization, object detection, and instance segmentation [18].

Object detection generalizes this to multiple objects, simultaneously detecting and classifying each instance in the scene using bounding boxes. This enables understanding of more complex environments where objects like **CAT**, **DOG**, and **DUCK** appear together. Instance segmentation goes a step further by assigning a pixel-level mask to each object, allowing for precise shape delineation and improved scene interpretation.

2.3 Object Tracking

Multi-object tracking (MOT) involves detecting and associating objects across video frames to maintain consistent identities [19]. Two main approaches dominate

2. Background Theory

the field: tracking by detection and joint detection and tracking (JDT).

Tracking by Detection separates detection and tracking into two stages. An object detector first identifies objects in each frame, followed by data association using motion models or appearance features. A leading method in this category is BotSORT [20], which improves robustness by associating both high-confidence and low-confidence detections.

Joint Detection and Tracking (JDT), or tracking-by-query, unifies detection and tracking into a single model. Transformer-based methods like DETR [21] and TrackFormer [22] use self-attention to model long-range temporal dependencies, enabling end-to-end reasoning about object appearance and motion. While these models achieve high accuracy, their quadratic complexity with respect to object count leads to high computational cost and memory usage, limiting their deployment in real-time settings. JDT models offer strong performance but are resource-intensive. In contrast, tracking-by-detection methods like BotSORT are more efficient and practical for real-time and industrial applications.

2.4 YOLOv11 Architecture for Backbone

The YOLOv11 architecture represents a significant advancement in real-time object detection and segmentation, combining improved accuracy with enhanced speed. One of the key innovations introduced in YOLOv11 is the C3K2 block—a specialized module that refines feature extraction while reducing computational overhead. Figure 2.2 illustrates the overall architecture, while Figure 2.3 focuses on the structure of the C3K2 block.

At its core, the YOLOv11 network follows the traditional design of having a backbone for feature extraction, a neck for feature aggregation, and a head for producing final predictions. However, YOLOv11 incorporates several modifications to improve performance. A central component is the C3K2 block, which builds upon the idea of Cross-Stage Partial (CSP). The C3K2 block is engineered to promote better gradient flow and to reduce redundancy in feature maps. It achieves this by combining a series of convolutional layers with efficient skip connections. The C3K2 block works by splitting the incoming feature map into two parts. One part is processed through a sequence of convolutions and bottleneck layers, while the other bypasses these layers. The outputs of both paths are then concatenated and further

2. Background Theory

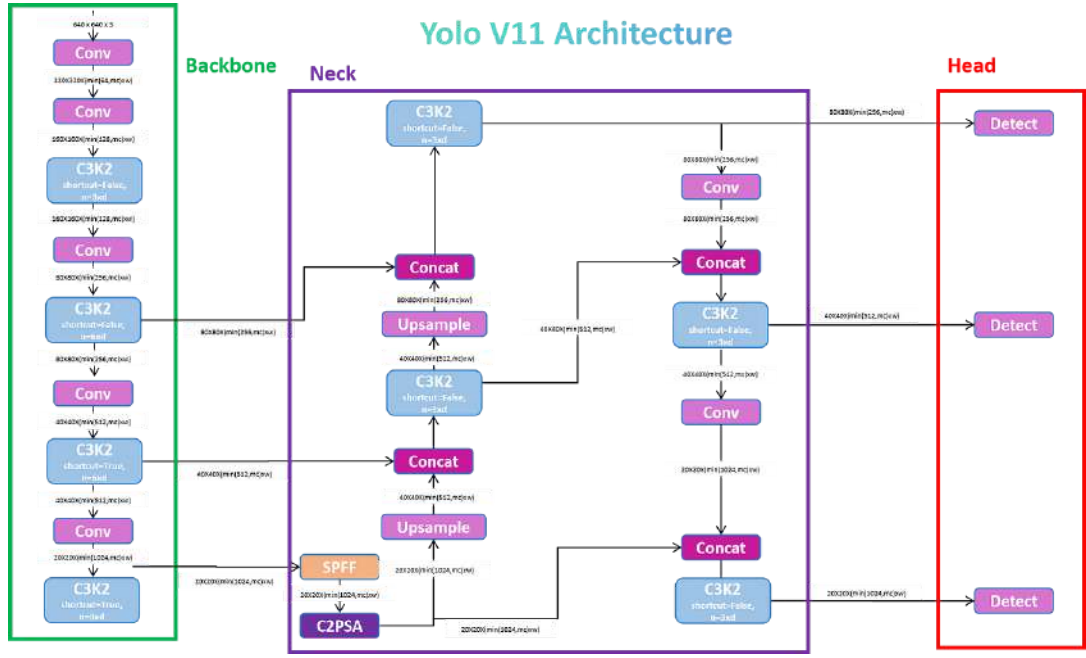


Figure 2.2: Overall architecture of YOLOv11 [23].

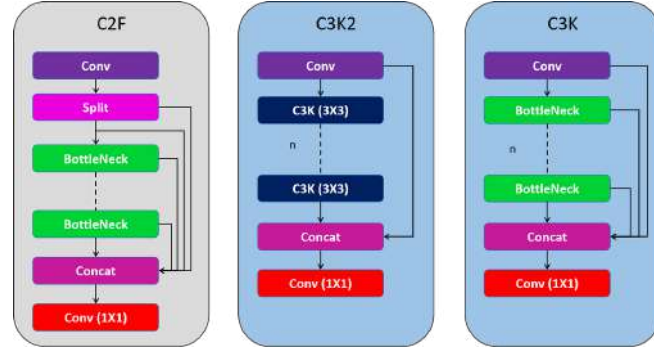


Figure 2.3: Detailed structure of the C3K2 block [23].

processed to produce a refined feature representation. This can be mathematically represented as: $\mathbf{y} = g(\mathbf{x}_1) \oplus \mathbf{x}_2$, where $\mathbf{x}$ is the input feature map, which is split into $\mathbf{x}_1$ and $\mathbf{x}_2$, $g(\cdot)$ denotes the sequence of convolutional and bottleneck operations, and $\oplus$ indicates channel-wise concatenation. This design minimizes information loss and improves the learning of complex features.

In addition to the C3K2 block, YOLOv11 employs other innovations such as enhanced skip connections and multi-scale feature aggregation modules. These enhancements work together to improve both detection and segmentation tasks, as the network can more accurately localize objects and delineate their boundaries in a single forward pass.

2.5 BotSORT for Multi-Object Tracking

BotSORT [20] extends SORT-style trackers by fusing *geometry*, *deep appearance cues*, and *camera-motion compensation*, achieving fewer identity switches and greater robustness in cluttered scenes. Incoming detections (e.g., from YOLO) pass through a **two-stage association**: A fast IoU match links high-confidence boxes to existing tracklets, and Unresolved detections are re-matched via cosine similarity of deep ReID embeddings [24, 25].

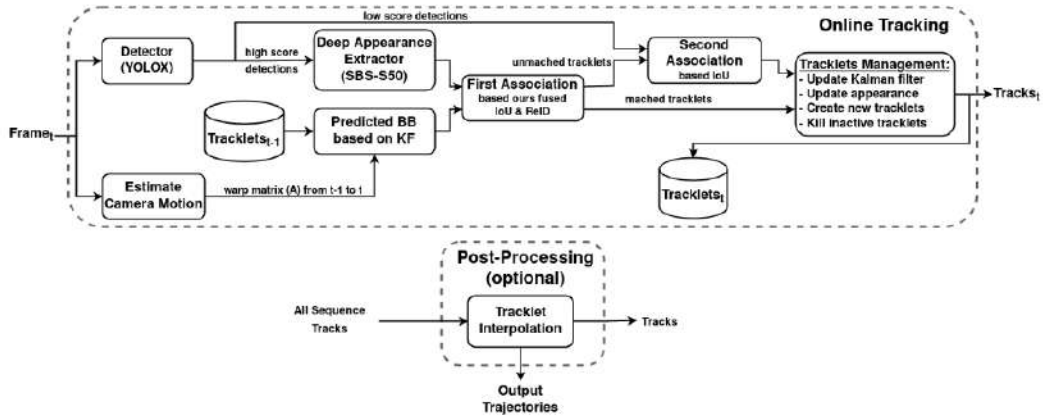


Figure 2.4: BotSORT pipeline: detections undergo IoU matching, ReID re-matching, egomotion projection, and adaptive tracklet management [20].

A homography-based egomotion module [26] projects tracklets into the current frame, preserving continuity during camera pans or tilts. Tracklets are activated only after persistent observations and discarded after a short untracked period, balancing false positives and latency. Details implementation of BotSORT is shown in Figure 2.4. Compared with SORT [27] and DeepSORT [1], this blend of spatial and appearance information delivers real-time, identity-consistent tracking across dynamic viewpoints.

2.6 Evaluation Metrics

Due to page limitations, the detailed evaluation metrics including MOTA, macro-MOTA, MOTP, Precision, Recall, and IDF1 are presented in the Appendix A.

Chapter 3

Literature Review

This chapter reviews the state-of-the-art research in sim2real learning, focusing on synthetic dataset creation, domain randomization, transfer strategies, and object tracking pipelines. A critical synthesis of these contributions is provided to establish the context for this work and identify current research gaps.

3.1 Synthetic Data Generation

A foundational contribution in this domain is the work of Tobin et al. [7], who introduced domain randomization as a means to close the sim2real gap. By randomizing visual parameters—such as textures, lighting, and camera poses—during training, models are encouraged to rely on essential object features rather than overfitting to consistent visual patterns. Their application to robotic grasping demonstrated a significant sim2real generalization capability.

Gaidon et al. [28] approached synthetic data generation differently through the “Playing for Data” framework. Using photorealistic video game engines, their system extracted pixel-perfect semantic annotations, which facilitated high-quality training for visual tasks such as segmentation and detection in real-world driving environments. Their results underlined the value of realism and environmental diversity in simulations.

Peng et al. [29] further extended the paradigm by introducing dynamics randomization. Instead of visual variation, they manipulated physical properties like mass, inertia, and friction in simulation to ensure robust control policies transferable to real-world robotics, especially in manipulation tasks.

3. Literature Review

More recent work by Dong et al. [30] presented a digital twin framework that leverages both synthetic and real data in a hybrid training setup. Their pipeline proved highly effective for robotic motion planning tasks, reaffirming the practicality of combining digital twins with transfer learning.

Ulhas and Kannapiran [31] applied Generative Adversarial Networks (GANs) [32] for visual domain adaptation in autonomous driving. By translating synthetic scenes into more realistic representations, their method reduced the domain gap and improved performance in real-world conditions.

Compared to domain adaptation, domain randomization (DR) offers several advantages, especially in robotics. While domain adaptation (DA) typically requires access to real-world data during training—often with costly or incomplete annotations—DR operates entirely in simulation and emphasizes robustness through controlled variability. This makes DR more scalable and data-efficient, especially for real-world tasks where collecting diverse and labeled real data is challenging. Moreover, DR promotes generalization by exposing models to a wide range of conditions they may face post-deployment, without requiring a specific target domain during training.

3.2 Empirical Benchmarks

Recent advances in sim2real transfer have significantly expanded the use of synthetic data for robotic perception and control. Early work in domain randomization (DR) demonstrated that heavy visual perturbations in simulation can yield models that generalize to the real world without any real-data fine-tuning. Tobin et al. [7] achieved 1.5 cm localization accuracy by randomizing textures and lighting in a grasping task, while Sadeghi and Levine’s CAD2RL framework used fully randomized 3D UAV simulations to obtain 100% collision avoidance in real flight [33]. Tremblay et al. [34] extended DR to vehicle detection, reaching 98.5% AP50 after fine-tuning with just 6K real images, and later employed GAN-based background fusion to attain 0.74 mAP on COCO validation [35].

Building on these randomized-only approaches, several works have combined DR with policy or representation learning. James et al. [36] trained pick-and-place policies via RL in randomized depth-color simulations, achieving 80% success on real robots. Mehta et al. [37] introduced differentiable simulation for visual servoing,

3. Literature Review

halving tracking loss compared to non-differentiable baselines. Horváth et al. [38] applied DR to RGB object detection, reporting 97.34% mAP50, and Zhang et al. [39] used texture-domain translation for mobile-robot navigation, improving real-world success rates by 35%. More recently, Horváth et al. [40] combined texture and lighting randomization with a Multi-Object Grasp Pose Estimation (MOGPE) to achieve up to 96.67% success in high-precision grasping.

In parallel, hybrid and self-supervised adaptation methods have begun to bridge the gap between pure DR and direct domain alignment. Ji et al. [41] proposed a Real2Sim2Real loop for continuum-robot control, integrating Koopman-operator models with multi-agent RL and safety-filtered online updates to cut free-space RMSE by 89.16%. Lu et al. [42] leveraged differentiable rendering for self-supervised camera-to-robot pose estimation, reaching sub-millimeter accuracy at 30 FPS without manual labels. Zhu et al. [43] showed that a ViT pretrained on a synthetic, domain-randomized parts dataset could classify real industrial parts with 90.91% accuracy. Ma et al. [44] introduced a global-to-local feature alignment via prototype matching for RGB-D grasp detection, boosting GraspNet-Planar AP by 6.5% under unsupervised adaptation. Complementary to these, Chen et al. [45] trained a domain-adaptive Faster R-CNN on 3K real driving images (38.97% AP50), and Sankaranarayanan et al. [46] used GAN-based adaptation to achieve 37.1% mIoU in segmentation with 5K unlabeled real images.

In this thesis, we fill this gap by integrating YOLOv11 with BotSORT for robust MOT. Our approach achieves 79.9% MOTA in a zero-shot transfer and 94.0% MOTA in a one-shot fine-tuning regime, outperforming existing sim2real methods while maintaining low computational overhead. To provide a consolidated comparison of these contributions, Table 3.1 summarizes key metrics and techniques from selected foundational and recent works. This table highlights the diversity of applications—from detection to Reinforcement Learning (RL) and the broad range of reported performance outcomes across tracking and domain randomization systems.

3.3 Research Gaps and Summary

Despite numerous advancements, several research challenges remain unresolved:

- Most sim2real studies focus on object detection; No prior research investigates

3. Literature Review

Table 3.1: Key Studies in Sim2Real

Author(s) & Year	Focus Area	Application	Technique	Reported Metrics
Tobin et al. (2017) [7]	DR	Robotic Grasping	Randomized textures, lighting	Success Rate: 97%
Tremblay et al. (2018) [34]	DR	Vehicle Detection	Synthetic dataset with varied backgrounds	AP ₅₀ : 98.5% (6K real fine-tune)
Chen et al. (2022) [45]	DA	Self-driving Detection	Domain-adaptive Faster R-CNN	AP ₅₀ : 38.97% (3K real)
Sadeghi & Levine (2017) [33]	DR + Policy Learning	UAV Flight Control	Domain-randomized 3D simulation	100% collision avoidance
James et al. (2019) [36]	DA (RCAN)	Pick & Place	RL with randomized depth/color/textures	80% success rate(real robot)
Mehta et al. (2022) [37]	DiffSim	Visual Servoing	Differentiable sim2real training	Tracking Loss ↓ 45%
Horváth et al. (2022) [38]	DR	Object Detection	Randomized texture, camera and object position, and additional image noise	Zero-shot: 86.32% mAP ₅₀ , One-shot: 97.38% mAP ₅₀
Zhang et al. (2023) [39]	DA (Texture Translation)	Mobile Robots	Texture-domain translation	Real-world success rate ↑ 35%
Tremblay et al. (2019) [35]	DA (GAN)	Object Detection	GAN-based background fusion	mAP ₅₀ : 74% (COCO val)
Horváth et al. (2023) [40]	DR	Pick and Place	randomized texture, object orientation, and additional image noise	Success Rate: 80% (RT), 96.67% (HP)
Ma et al. (2024) [44]	DA (Proto-Match)	RGB-D Grasp Detection	Global-to-local alignment with prototype matching	GraspNet-Planar: +6.5% AP improvement
Zhu et al. (2023) [43]	DR	Industrial Parts Classification	Synthetic dataset with domain randomization	Best Accuracy: 90.91% (ViT, Syn-R)
Lu et al. (2023) [42]	Self-Sup. Transfer	Camera-to-Robot Pose Estimation	Differentiable rendering with self-supervision	AUC: 91.2%, Speed: 30 FPS
Ji et al. (2024) [41]	Real2Sim2Real	Continuum Robot Control	Koopman operator modeling, MARL with online update and safety QP	RMSE ↓ 89.16% (Free), ↓ 85.70% (Payload), Sub-mm Accuracy
Our Results	DR	MOT	Photometric, Occlusion, Background & Lighting	MOTA: 79.9% (Zero Shot), 94% (One Shot)

3. Literature Review

tracking. This thesis is the first to establish a sim2real benchmark for multi-object tracking using synthetic-only training.

- Synthesizes fully dynamic video sequences with Poisson-disk sampling for non-overlapping layouts (and controlled overlaps for occlusions) and rich photometric randomization (lighting, hue, HSVA color shifts)
- Trains a joint detection–tracking model under both zero-shot and one-shot real-data scenarios.
- Rigorously evaluates temporal identity metrics (MOTA, IDF1) in robotic settings.
- Comparison of Detection based and Segmentation-Based tracking.

This thesis addresses these gaps by proposing a detection-based object tracking framework enhanced by domain randomization and evaluated in both zero-shot and one-shot transfer scenarios. The framework is experimentally validated on real-world video sequences, highlighting its applicability to robotic systems requiring efficient and scalable perception solutions.

Chapter 4

Methodology

In this chapter, we describe the methodology adopted to develop and evaluate our sim2real transfer learning framework for object tracking in robotics. This chapter outlines the detailed procedures, tools, and protocols employed in our research, establishing a reproducible framework for assessing the performance of our object tracking system in a robotics context.

4.1 Sim2Real Knowledge Transfer

Sim2real knowledge transfer focuses on training models with synthetic data and then deploying them in real-world environments, thereby “transferring” learned representations across domains. In our context, the synthetic domain (D_S) consists of computer-generated images derived from high-fidelity 3D models, while the real domain (D_T) includes actual images captured from robotic scenarios. Despite the advantages of scalability and controllability in simulation, a notable gap often called the reality gap, can emerge when a model trained solely on synthetic data is applied to real-world tasks.

Figure 4.1 offers a concise overview of how we manage sim2real knowledge transfer within our system. The left part outlines the creation of synthetic training datasets using 3D models and advanced domain randomization. The resulting model is then evaluated using a real-world test set, where manually annotated videos serve as the ground truth.

Formally, let us define a domain $D = \{\mathcal{X}, P(\mathbf{x})\}$ with a feature space $\mathcal{X}$ and a probability distribution $P(\mathbf{x})$. The source domain $\{D_S, T_S\}$ corresponds to the syn-

4. Methodology

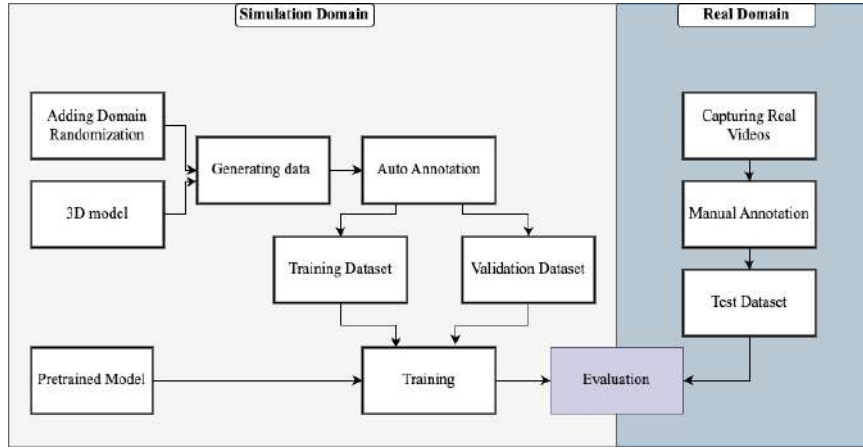


Figure 4.1: Overview of sim2real knowledge transfer. Adopted to MOT from [38]

thetic dataset and object tracking task, while the target domain $\{D_T, T_T\}$ comprises real images of the same task. Our goal is to learn a predictive function $f_T(\cdot)$ that performs well on $\{D_T, T_T\}$ by leveraging the knowledge gained from $\{D_S, T_S\}$. In particular, we investigate:

- **Zero-Shot Transfer Learning**, where the model trained on synthetic data is directly evaluated on real data without any fine-tuning.
- **One-Shot Transfer Learning**, where a small amount of real data (e.g., one annotated image per class) is introduced during training to refine the model.

By comparing performance across these two scenarios, we quantify how effectively the model generalizes to real-world conditions and to what extent minimal real data can mitigate the reality gap. This sim2real transfer framework underpins the broader methodology presented in this thesis, guiding how synthetic and real datasets are utilized for object tracking in robotics.

4.2 Synthetic data generation

4.2.1 Domain Randomization Techniques

To ensure that our synthetic dataset captures a wide range of visual variations and mimics the complexity of real-world environments, we employ several domain randomization techniques. These randomizers are designed to vary key aspects of the scene, such as object placement, color properties, rotation, lighting conditions,

4. Methodology

and textures. In this section, we describe each randomizer used in our pipeline, highlighting both the implementation details and the underlying theoretical principles.

Object Placement

The Object Placement uses Poisson Disk Sampling [47] to generate a set $S = \{(x_i, y_i)\}$ of non-overlapping points in a $w \times h$ area with a random seed, satisfying

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq d \quad \forall i \neq j,$$

where d is the minimum separation distance. An offset is applied to center the layout, and a fixed depth value is assigned to position objects in 3D space (see Algorithm 1). This technique avoids object clustering and encourages spatial diversity. This technique avoids object clustering and encourages spatial diversity.

Algorithm 1 Object Placement via Poisson Disk Sampling

```

1: procedure PLACEOBJECTS(minDist, seedProvider)
2:                                      $\triangleright$  Generate non-overlapping sample points
3:   seed  $\leftarrow$  seedProvider.nextSeed()
4:   samples  $\leftarrow$  PoissonDiskSampling(width, height, minDist, seed)
5:   offset  $\leftarrow$   $(-\frac{1}{2} \cdot \text{width}, -\frac{1}{2} \cdot \text{height}, 0)$ 
6:   for all  $(x, y) \in \text{samples}$  do
7:     proto  $\leftarrow$  prototypes.sampleRandom()
8:     obj  $\leftarrow$  pool.getOrCreate(proto)
9:     obj.position  $\leftarrow$   $(x, y, \text{depth}) + \text{offset}$ 
10:    container.add(obj)
11:   end for
12: end procedure

```

Rotation Randomizer

At the start of each iteration, all objects matching a given filter are assigned a random orientation. For each object, three Euler angles are drawn independently from the uniform distribution on $[0^\circ, 360^\circ]$,

$$\alpha, \beta, \gamma \sim \mathcal{U}(0^\circ, 360^\circ),$$

converted to a rotation in Quaternion space (see Algorithm 2) and applied to the object. This ensures robustness to viewpoint changes.

4. Methodology

Algorithm 2 Rotation Randomizer

```

1: procedure ONITERATIONSTART(tagFilter)
2:   objects  $\leftarrow$  QueryObjects(tagFilter)
3:   for all obj  $\in$  objects do
4:      $\alpha \leftarrow \text{Uniform}(0^0, 360^0)$ 
5:      $\beta \leftarrow \text{Uniform}(0^0, 360^0)$ 
6:      $\gamma \leftarrow \text{Uniform}(0^0, 360^0)$ 
7:      $q \leftarrow \text{EulerToQuaternion}(\alpha, \beta, \gamma)$ 
8:     obj.orientation  $\leftarrow q$ 
9:   end for
10: end procedure

```

Hue Offset Randomizer

The Hue Offset Randomizer perturbs the perceived color of each entity by adding a random offset to its hue channel. This models natural variations—such as material aging or lighting shifts—and encourages the network to focus on shape and structure rather than specific color tones. At each iteration, every entity matching a given filter tag receives an independent hue shift

$$\theta \sim \mathcal{U}(\theta_{\min}, \theta_{\max}),$$

which is applied via its material or shader parameter (see Algorithm 3). The hue offset randomizer assigns each entity a new hue value θ drawn uniformly from $[\theta_{\min}, \theta_{\max}]$. Since hue values are defined in the circular range $[0^\circ, 180^\circ]$, any values outside this range are wrapped around using modulo 180° to maintain continuity in the hue space.

Algorithm 3 Hue Offset Randomizer

```

1: procedure ONITERATIONSTART(tagFilter,  $\theta_{\min}$ ,  $\theta_{\max}$ , hueProp)
2:   entities  $\leftarrow$  FindEntitiesWithTag(tagFilter)
3:   for all entity in entities do
4:      $\theta \leftarrow \text{Uniform}(\theta_{\min}, \theta_{\max})$ 
5:     SetMaterialParameter(entity, hueProp,  $\theta$ )
6:   end for
7: end procedure

```

Color Randomizer

The Color Randomizer replaces each object’s base material color with a new random color sampled in HSVA space. This covers variations in hue, saturation, and

4. Methodology

brightness simultaneously, enabling the model to generalize across diverse object appearances (see Algorithm 4).

Algorithm 4 Color Randomizer

```

1: procedure ONITERATIONSTART(tagFilter, colorSampler)
2:   objects  $\leftarrow$  QueryObjects(tagFilter)
3:   for all obj in objects do
4:     c  $\leftarrow$  colorSampler.Sample()
5:     SetMaterialColor(obj, "BaseColor", c)
6:   end for
7: end procedure

```

Lighting Randomizer

Algorithm 5 Lighting Randomizer

```

1: procedure ONITERATIONSTART(lightsList, hourRange, dayRange, latRange)
2:   h  $\leftarrow$  Uniform(hourRange.min, hourRange.max)
3:   d  $\leftarrow$  Uniform(dayRange.min, dayRange.max)
4:   l  $\leftarrow$  Uniform(latRange.min, latRange.max)

5:                                      $\triangleright$  Time-of-day rotation (Earth spin)
6:   spin  $\leftarrow$  AngleAxis( $\frac{h+12}{24} \times 360^\circ$ , (0, -1, 0))

7:                                      $\triangleright$  Seasonal tilt (Earth axial tilt)
8:    $\phi \leftarrow \frac{d}{365} \times 360^\circ$ 
9:   tilt  $\leftarrow$  Euler( $\cos \phi \cdot 23.5^\circ$ , 0,  $\sin \phi \cdot 23.5^\circ$ )

10:                                      $\triangleright$  Latitude adjustment
11:   latRot  $\leftarrow$  AngleAxis(l, (1, 0, 0))

12:                                      $\triangleright$  Compose world-space sun rotation
13:   sunRot  $\leftarrow$  tilt  $\times$  spin  $\times$  latRot

14:                                      $\triangleright$  Align noon direction downward
15:   align  $\leftarrow$  Euler(90°, 0, 0)
16:   finalRot  $\leftarrow$  align  $\times$  Inverse(sunRot)

17:   for all light in lightsList do
18:     SetRotation(light, finalRot)
19:   end for
20: end procedure

```

To introduce realistic lighting conditions into the synthetic scenes, we implement a Lighting Randomizer that simulates natural variations caused by time of day, seasonal changes, and geographic location. At each simulation step, it randomly samples the hour, day of the year, and observer latitude to compute a corresponding sun orientation. This dynamic adjustment mimics the Earth’s axial tilt, daily rotation, and

4. Methodology

position on the globe, ensuring that objects are illuminated from varied angles across scenes. The computed light direction is then applied uniformly to all light sources in the environment, enabling the generation of diverse and photorealistic lighting conditions that improve generalization for real-world deployment. The full procedure is described in Algorithm 5. This enables the rendering engine to generate varied light angles, intensities, and shadow distributions.

Background Texture Randomizer

The background texture randomizer diversifies background surface appearance by randomly selecting and assigning textures from a predefined catalogue [48, 49, 50] to each target entity (see Algorithm 6). This simulates background material (e.g. table) and environmental variation (e.g. wear, dirt, pattern changes) and improves model robustness to background texture changes.

Algorithm 6 Background Texture Randomizer

```

1: procedure ONITERATIONSTART(targetsList, texturesList, propertyName)
2:   for all entity  $\in$  targetsList do
3:      $\tau \leftarrow \text{UniformCategorical}(\textit{texturesList})$ 
4:     SetBackgroundMaterialTexture(entity, propertyName,  $\tau$ )
5:   end for
6: end procedure

```

Overlapping Object Randomizer

To generate cluttered foreground layouts we keep a small set of fixed *anchor transforms* $\mathcal{A} = \{(\mathbf{p}_j, \mathbf{r}_j)\}_{j=1}^{N_A}$, where $\mathbf{p}_j \in \mathbb{R}^3$ is a position and $\mathbf{r}_j \in \text{SO}(3)$ a rotation and N_A is the total number of predefined anchors. At every simulation step we draw a user-specified number $N_{\text{new}} \leq N_A$ of objects and assign them to anchors *with replacement*. If an anchor is selected multiple times within the same step, each successive object is translated backwards along the positive z -axis by an integer multiple of a constant offset Δz . Thus the objects' image projections overlap almost perfectly, yet their 3-D volumes are never closer than Δz (see Algorithm 7). This produces realistic partial occlusions and depth-ordering variations, teaching the model to maintain accurate detection and association under heavy object overlap.

This strategy assumes a fixed camera pose in which the camera's optical axis is approximately aligned with the global z -axis. Under this configuration, translating

4. Methodology

objects along the z -axis causes their 2D image projections to overlap meaningfully. If the camera were rotated significantly, this spatial logic would not yield overlapping projections in image space.

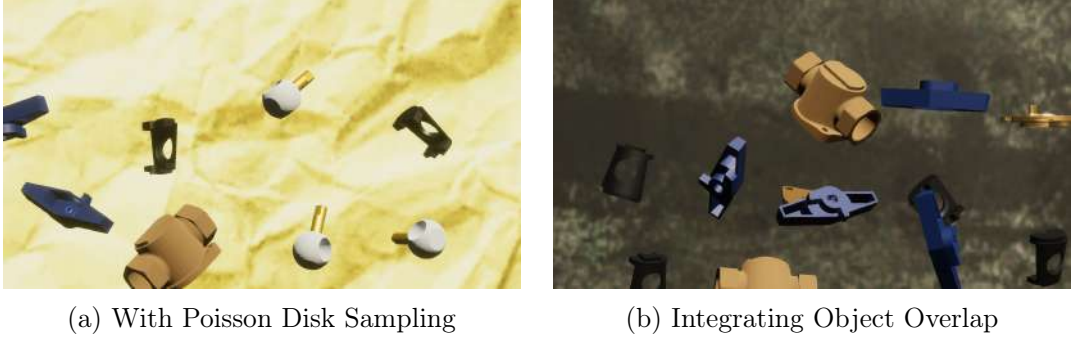


Figure 4.2: Comparison of placing objects

Algorithm 7 Object Overlapping Placement

```

1: procedure PLACEANCHORED OBJECTS( $prototypes, \mathcal{A}, N_{\text{new}}, \sigma, \Delta z$ )
2:    $C \leftarrow \{c_j \leftarrow 0 \mid j = 1, \dots, |\mathcal{A}|\}$  ▷ Reuse counters, reset each step
3:   for  $i = 1$  to  $N_{\text{new}}$  do
4:      $(\mathbf{p}, \mathbf{r}) \leftarrow \text{Uniform}(\mathcal{A})$ 
5:      $j \leftarrow \text{index of chosen anchor}$ 
6:      $c \leftarrow C[j]$  ;  $C[j] \leftarrow c + 1$ 
7:      $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$ 
8:      $proto \leftarrow \text{UniformCategorical}(prototypes)$ 
9:      $obj \leftarrow \text{Instantiate}(proto)$ 
10:     $obj.\text{position} \leftarrow \mathbf{p} + \boldsymbol{\varepsilon} + (0, 0, c \cdot \Delta z)$ 
11:     $obj.\text{rotation} \leftarrow \mathbf{r}$ 
12:  end for
13: end procedure

```

4.2.2 Camera Setup and Capture

Our synthetic data generation pipeline is built in Unity, utilizing the *PerceptionCamera*, an extension of Unity’s native camera system, to capture RGB images and automatically generate annotations (See Algorithm 10). The camera is configured with custom parameters such as resolution, field of view (FOV), and a fixed frame rate, ensuring consistent simulation timing.

During each frame’s rendering phase, the camera captures the scene view and encodes it in PNG format. Object positions are projected from 3D world coordinates to 2D image space using the camera’s projection matrix. For any 3D point $\mathbf{p} =$

4. Methodology

$[X, Y, Z, 1]^T$, the projection follows:

$$\mathbf{p}' = \frac{P \cdot \mathbf{p}}{w},$$

where P is the 3×4 projection matrix and w is the homogeneous scaling factor. The projected 2D coordinates are used to compute axis-aligned bounding boxes (AABBs) (See Algorithm 8), forming the ground truth for object tracking.

The combination of automated image capture, intrinsic-aware annotation, and structured domain randomization forms the foundation of our synthetic dataset. By systematically varying scene-level and object-level features, the pipeline creates a diverse set of training examples. These are critical to achieving robust sim2real transfer for object tracking in robotic environments.

Algorithm 8 Compute Axis-Aligned Bounding Box (AABB)

```

1: procedure COMPUTEAABB( $\{(X_i, Y_i, Z_i)\}$ ,  $P$ ,  $(W, H)$ , origin)
2:   initialize lists  $U$ ,  $V \leftarrow []$ 
3:   for all  $(X, Y, Z)$  in Vertices do
4:      $\mathbf{p}_h \leftarrow P \cdot [X, Y, Z, 1]^T$ 
5:      $u \leftarrow \mathbf{p}_h.x / \mathbf{p}_h.w$ ,  $v \leftarrow \mathbf{p}_h.y / \mathbf{p}_h.w$ 
6:     append  $u$  to  $U$ , append  $v$  to  $V$ 
7:   end for
8:    $x_{\min} \leftarrow \min(U)$ ,  $x_{\max} \leftarrow \max(U)$ 
9:    $y_{\min} \leftarrow \min(V)$ ,  $y_{\max} \leftarrow \max(V)$ 
10:  return  $(x_{\min}, y_{\min}, x_{\max} - x_{\min}, y_{\max} - y_{\min})$ 
11: end procedure

```

4.3 Training

To train the model using the synthetic dataset generated with our method, a series of preprocessing steps were conducted to prepare the data for both instance segmentation and object detection-based tracking tasks.

For the segmentation task, the Unity SOLO-formatted data was first converted to the COCO format using the `pysolotools` library [51]. Subsequently, the COCO-format RLE (Run-Length Encoding) segmentation annotations were transformed into YOLO polygon format. This was accomplished by decoding the RLE masks into binary masks, extracting contours using OpenCV [52], normalizing the resulting

4. Methodology

polygon coordinates with respect to the image dimensions, and writing the annotations into YOLO-compatible label files, one per image.

For the object detection task, the Unity SOLO format was converted to the YOLO format by iterating through each sequence’s image and corresponding JSON annotation file. Bounding box information was extracted, and the coordinates were normalized based on image width and height. Class labels were mapped to zero-indexed YOLO class IDs, and the annotations were saved in the standard YOLO format alongside the corresponding images.

These preprocessing steps ensured compatibility with YOLO-based models and enabled robust training using the synthetic data generated through Unity Perception.

4.3.1 Training Protocol

Even though our sim2real framework can be adapted to various CNN-based object detectors, we have opted for YOLOv11 due to its balanced trade-off between accuracy and speed. In addition, YOLOv11 natively supports an advanced data augmentation pipeline, which is crucial for achieving robust performance when transferring from synthetic to real-world domains.

For the training process, we initialize YOLOv11 with weights pre-trained on COCO, ensuring that the early layers already capture fundamental visual features such as edges and textures. This approach accelerates convergence and often yields better performance than training from scratch.

Table 4.1 provides an overview of these augmentations. Notably, we preserve the original parameter names (e.g., `angle`, `saturation`, `exposure`, `hue`, `mosaic`, `jitter`, `random`) for clarity and consistency with existing YOLO frameworks.

Table 4.1: Data Augmentation Techniques for Training YOLOv11

Name	Description
Angle	Randomly rotates images.
Saturation	Randomly changes the saturation.
Exposure	Randomly changes the brightness.
Hue	Randomly changes the hue color channel.
Mosaic	Uses four images in a single composite (Mosaic).
Jitter	Randomly resizes images and adjusts aspect ratio.
Random	Randomly resizes network input after every ten batches.

These augmentations introduce diverse variations in object appearance, lighting, and geometry, thereby simulating real-world conditions more accurately. During training, each mini-batch is sampled from the synthetic dataset (and, in the one-shot case, supplemented with a small number of real images), with augmentations applied randomly. We use AdamW [53] with momentum (0.9) for optimization, while a suitable learning rate schedule (0.001111) is adopted to balance fast convergence and stable training.

By combining the YOLOv11 architecture, COCO-pretrained weights, and advanced data augmentation, we obtain a detector that is both efficient and capable of generalizing effectively to real-world scenarios addressing the sim2real gap in object tracking tasks.

4.4 Multiple Object Tracking

YOLOv11 was integrated with BotSORT to enable real-time multi-object tracking across frames. The detector provides high-confidence bounding boxes, which are first associated with existing tracklets using Intersection over Union (IoU). Unmatched detections and tracklets are then processed using deep appearance features extracted via a ReID model (e.g., SBS-50 [20]) to resolve ambiguities caused by occlusion or proximity. Tracklet updates rely on both geometric and appearance-based matching, while unmatched tracks are terminated after a defined buffer period unless re-associated (see Figure 2.4). Although BotSORT supports camera motion compensation through homography-based warping, this feature was disabled in our setup due to the use of a static camera.

This integration leverages YOLOv11’s detection accuracy and BotSORT’s robust identity preservation to deliver stable tracking performance across complex scenes. As shown in Chapter 6, this setup significantly reduces identity switches and improves MOTA in both zero-shot and one-shot sim2real evaluations.

Chapter 5

Dataset

This chapter introduces the datasets used for evaluating the performance and robustness of the proposed object tracking framework. It covers both the real-world test data and the synthetic dataset created using high-fidelity 3D CAD models.

5.1 Annotated Test Data

To rigorously assess the tracking model under real-world conditions, we curated a test dataset comprising two manually annotated video sequences. These sequences capture diverse challenges such as partial occlusions, motion blur, complex object arrangements, and varying illumination settings.

Figure 5.1 highlights representative frames illustrating key difficulties present in the dataset: natural lighting variations, motion blur, hand-induced occlusions, and artificial lighting. These conditions simulate realistic deployment scenarios and help evaluate the robustness of the tracking framework under noisy visual input.

Annotations follow a modified MOTChallenge format and include frame-wise information such as unique object IDs, bounding box coordinates, class labels, and visibility flags. Each object is consistently labeled across frames with a class ID from 1 to 5.¹

Table 5.1 provides a summary of the annotated test sequences. Both videos span over a thousand frames and contain numerous uniquely tracked objects. Class labels are evenly distributed to support balanced class-wise performance evaluation.

¹The dataset is available at https://drive.google.com/drive/folders/1J190v0cuNuGJ2QLT51_hAIZTYgqK02td?usp=sharing.

5. Dataset

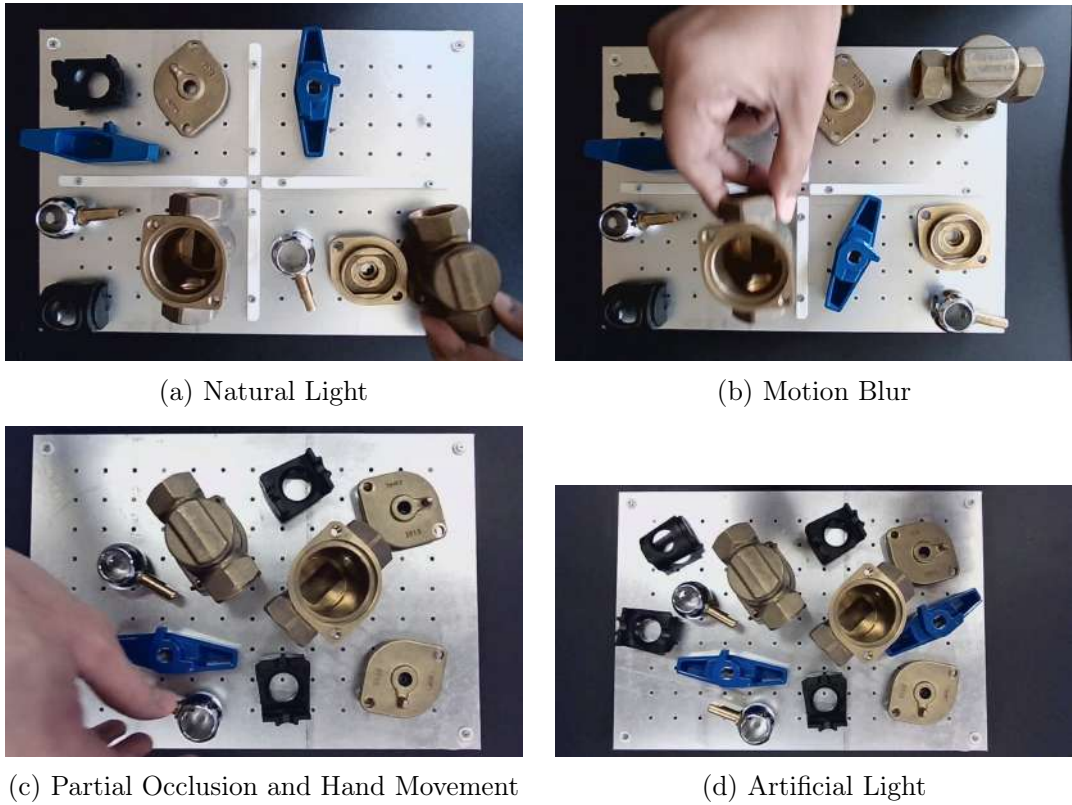


Figure 5.1: Challenging visual conditions present in the test dataset

Table 5.1: Summary of Annotated Test Videos

Video	Frames	Objects	Annotations	Classes
Video 1	1,426	40	13,959	5
Video 2	1,166	58	8,695	5

Figure 5.2 shows the number of annotations per class across both videos. This balanced distribution enables a fair analysis of the model’s generalization and inter-class discrimination capabilities in visually complex scenarios.

5.2 3D CAD Models for Synthetic Data Generation

To support the generation of synthetic training data, we developed five high-fidelity 3D CAD models using Blender, representing industrial components commonly encountered in robotic manipulation tasks: *handle*, *ball*, *body*, *bonnet*, and *seat*.

Four of the objects were modeled with a metallic appearance, while the seat was designed with a plastic-like surface. Each object was assigned a representative

5. Dataset

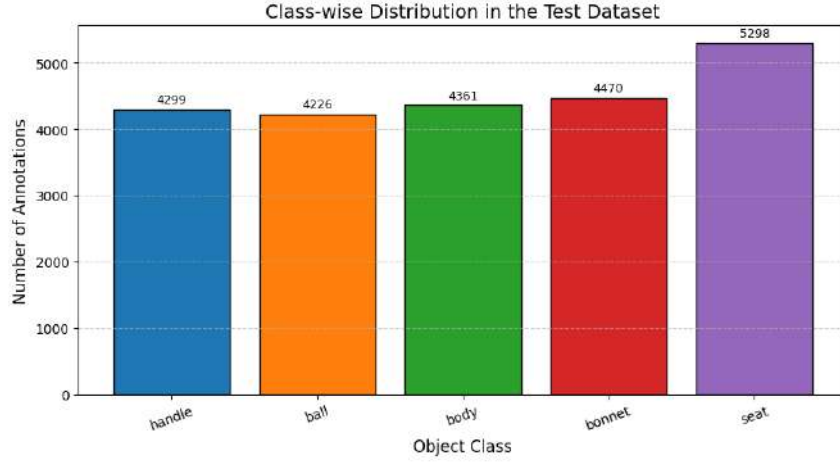


Figure 5.2: Class distribution across the combined test dataset.

HEX color code, reflecting real-world appearance to enhance realism. Table B.1 summarizes the material types and corresponding color codes.

Each model was precisely designed to capture accurate geometry, surface details, and reflectance properties using Blender’s sculpting and material tools. Representative labeled views of both synthetic (simulation) and real-world object appearances are shown in Figures 5.3

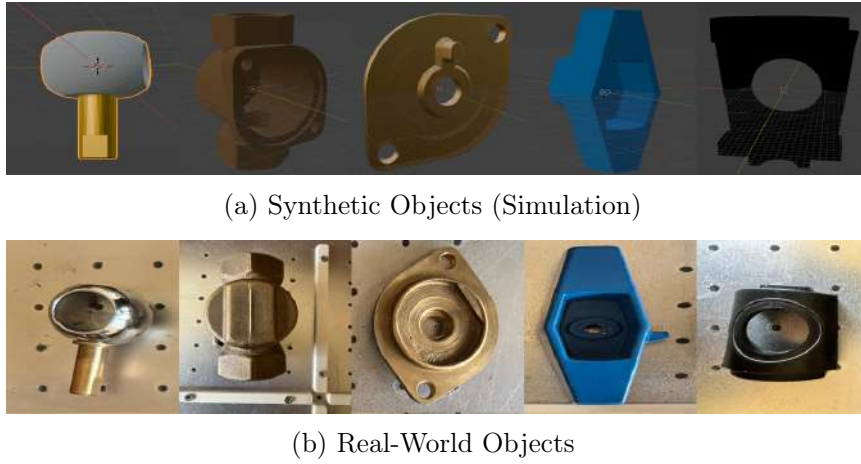


Figure 5.3: Visual comparison between simulation-based synthetic objects (top) and their corresponding real-world counterparts (bottom). All individual object images were resized to (256×256 pixels) for consistent visual alignment.

While the 3D CAD models described in this chapter are essential for benchmarking and evaluation, it is important to emphasize that the proposed tracking method itself is not dependent on any specific object design. The framework is designed to be applied to any object class.

Chapter 6

Results and Evaluation

This chapter presents the evaluation results of our sim2real transfer learning framework for object tracking in robotics. We assess model performance for multi-object tracking tasks using real-world test data.

Key metrics such as Precision, Recall, MOTA, MOTP and IDF1 are used to quantify performance. Both zero-shot and one-shot transfer learning scenarios are compared to analyze generalization capabilities. Qualitative examples are included to illustrate the model’s robustness in realistic conditions.

6.1 Dataset Variants and Randomization Composition

The success of sim2real transfer learning in object tracking largely depends on the quality and variability of the synthetic datasets. To systematically evaluate the effect of domain randomization components on model generalization, we generated a collection of synthetic datasets, each with controlled variation in visual attributes. These datasets were created using the methodology described in Section 4.2.1, leveraging procedural scene generation, randomized rendering parameters, and intrinsic annotation capabilities. All datasets share a foundational scene structure, with key parameters summarized in Table 6.1.

To investigate the influence of individual randomization strategies on sim2real transfer, we generated seven synthetic dataset variants (D1–D7), each sharing random object placement, rotation, textured backgrounds, and lighting but differing in occlusion and color perturbations presented in Table 6.2.

6. Results and Evaluation

Table 6.1: Domain Randomization Parameters Used Across Datasets

Parameter	Value
Object Placement Distance	1.5 units
Object Depth from Background	−1.5 units
Object Placement Area (x, y)	$(4, 8)$ units
Object Rotation (x, y, z)	$\mathcal{U}(0^\circ, 360^\circ)$
Background Texture	215 images
Overlapping Objects	0 or 2 or 5

Table 6.2: Randomization Techniques Across Synthetic Datasets

Dataset	Place	Rot.	Tex. BG	Light	Overlap	Color	Hue
D1	✓	✓	✓	✓	5	✗	✗
D2	✓	✓	✓	✓	2	✓	✓
D3	✓	✓	✓	✓	2	✗	✗
D4	✓	✓	✓	✓	✗	✗	✗
D5	✓	✓	✗	✗	2	✗	✗
D6	✓	✓	✓	✗	2	✗	✗
D7	✗	✓	✓	✓	✗	✗	✗

D1 combines heavy occlusion (five overlapping objects per frame) with full photometric randomization but omits color and hue shifts, serving as a baseline for extreme spatial clutter. D2 reduces overlap to two objects while adding both color randomization and hue-offset variations to assess chromatic diversity effects. D3 matches D2’s two-object overlap but removes color and hue perturbations, isolating the benefit of chromatic augmentation. D4 retains full photometric variability without any object overlap, isolating the impact of occlusion removal. D5 provides a low-variation baseline by applying only placement, rotation, and two-object overlap—disabling both background-texture and lighting randomization—to measure performance under minimal photometric diversity. D6 combines two-object overlap with background texture variation but omits lighting randomization to quantify illumination’s role in domain invariance. Finally, in D7, all settings remain identical to D4 except for the object placement strategy, where a uniform distribution in the range of -3 to 3 replaces the Poisson-disk sampling used in our method. Representative samples appear in Figure 6.1, and the configurations are summarized in Table 6.2.

6. Results and Evaluation

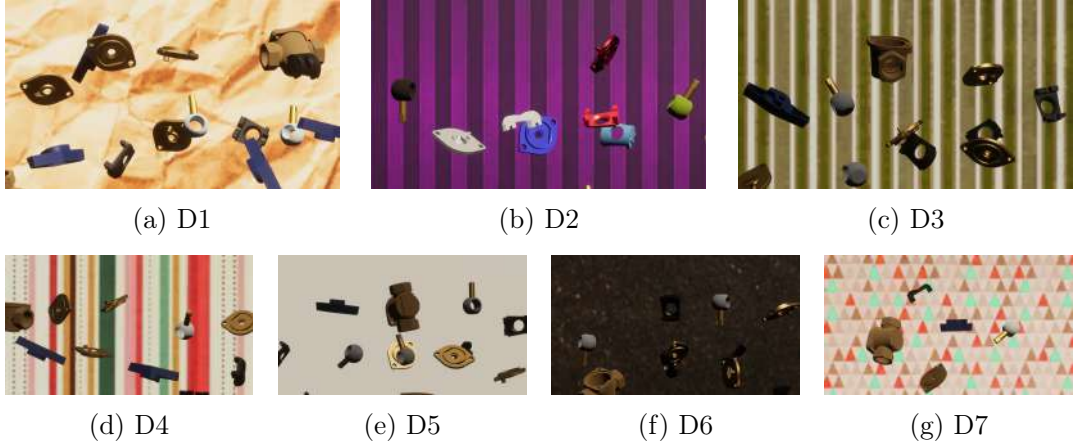


Figure 6.1: Examples of synthetic data generated across different datasets.

6.2 Evaluation Protocol

Experiments were conducted on real-world videos. Experiments were conducted on our benchmark detailed in Section 5.1. Each dataset variant was trained and evaluated across three independent runs to account for Standard Deviation (STD). Reported final results reflect the mean with STD across these runs. Performance was assessed using standard metrics including MOTA, Macro-MOTA, IDF1, Precision, Recall, and MOTP.

6.3 Detection-Based Tracking on Synthetic Domain

The object detection model was trained as a foundation for detection-based tracking using an NVIDIA RTX 3050 Mobile GPU. Early stopping with a patience of 10 epochs was employed to avoid overfitting. Key training settings and outcomes are presented in Table 6.3.

Table 6.3: Training Summary for Object Detection

Setting	Value
Epochs (max)	100
Early Stop Patience	10
Avg. Epochs Run	~70
Final mAP	~98.4%
Duration	~1.75 hrs
Stop Condition	Validation stagnation

The model typically converged within 70 epochs, achieving a final mAP@0.5 of approximately 98.4% in around 1.75 hours on the synthetic validation set. Figure 6.2

6. Results and Evaluation

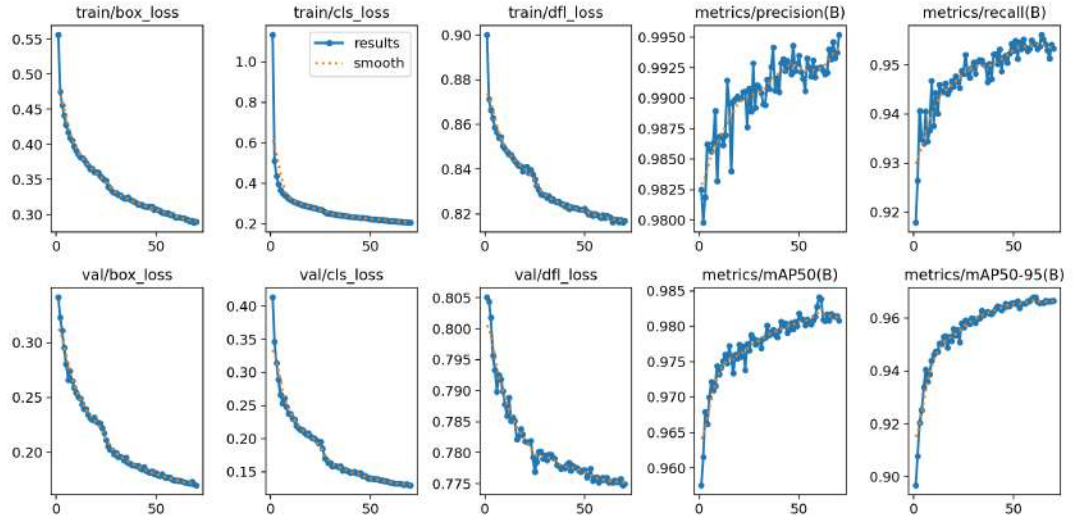


Figure 6.2: Training and validation loss, and evaluation metric trends over epochs.

illustrates the training and validation losses, along with metrics such as precision, recall, $mAP@0.5$, and $mAP@[0.5:0.95]$. The smooth decline in losses and steady rise in evaluation metrics indicate strong generalization without overfitting on the synthetic domain.

The F1-Confidence curve in Figure 6.3a highlights the model’s calibration. The overall F1 score peaks at 0.97 near a confidence threshold of 0.437. High per-class F1 scores—especially for *ball*, *handle*, and *bonnet*—demonstrate robust detection across classes. However, the F1 score drops sharply near a confidence of 1.0. This occurs because very few detections meet such a strict threshold, causing both recall and precision to fall dramatically. The use of overlapping object placement in the synthetic dataset likely amplifies this effect, as occlusions and depth ambiguities reduce the model’s confidence in otherwise correct detections.

The Precision-Recall curve in Figure 6.3b shows high performance across all classes, with precision and recall values often exceeding 0.99. The lowest $mAP@0.5$ is 0.944 for the *body* class, confirming strong overall detection performance ($mAP@0.5 = 0.984$).

6.4 MOT configuration

Integrating the YOLOv11 detector with the BotSORT algorithm is used with configuration shown in Table 6.4. A high association threshold (`track_high_thresh`

6. Results and Evaluation

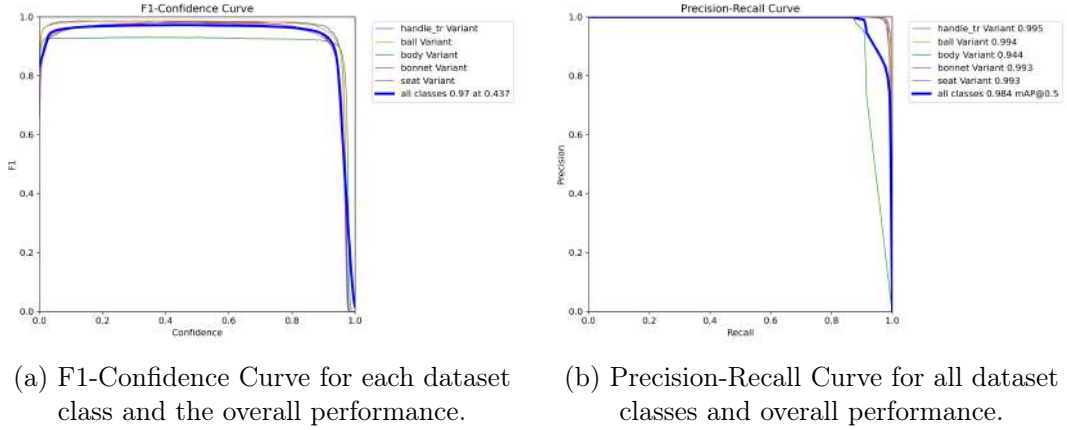


Figure 6.3: Model training evaluation

= 0.5) was used to ensure that only confident detections are linked to existing tracklets, while a lower threshold (`track_low_thresh` = 0.1) allowed temporary retention of less confident detections during occlusions. The `new_track_thresh` was set to 0.5 to enable new tracks even when detection confidence momentarily drops, enhancing robustness in cluttered scenes. A `track_buffer` of 60 ensured that identities persist during short occlusions. The `match_thresh` of 0.9 controlled how strictly new detections are matched to tracklets based on appearance and position. To avoid spurious matches, the `proximity_thresh` and `appearance_thresh` were set to 0.5 and 0.25, respectively. Re-identification (`with_reid` = true) was enabled to recover object identity after occlusion. Since all experiments used a static camera, geometric motion compensation (`gmc_method`) was disabled.

Table 6.4: BotSORT Tracker Configuration

Parameter	Value
<code>fuse_score</code>	false
<code>track_high_thresh</code>	0.5
<code>track_low_thresh</code>	0.1
<code>new_track_thresh</code>	0.5
<code>track_buffer</code>	60
<code>match_thresh</code>	0.9
<code>proximity_thresh</code>	0.5
<code>appearance_thresh</code>	0.25
<code>with_reid</code>	true
<code>gmc_method</code>	none

6.5 Detection-Based Tracking Performance

This section presents a thorough analysis of the detection-based tracking pipeline trained with seven different domain randomization configurations. All models were evaluated on the same real-world test dataset under a strict zero-shot condition, with no real image fine-tuning. Performance was assessed using standard tracking metrics, including MOTA, Macro-MOTA, MOTP, IDF1, and the number of ID switches.

We evaluate detection-based tracking on Video 2 (V2) test set using seven synthetic training configurations (Table 6.2). Table 6.5 reports our experimental results. All variants include placement, rotation, texture background, and lighting randomizers; we systematically ablate object overlap, color randomization, hue offset, and lighting to isolate their effects.

Occlusion (Object Overlap): Reducing overlap from five to two objects per scene yields a substantial improvement in MOTA. Dataset D1 achieves only 0.704 MOTA, whereas all two object overlap variants (D2, D3 and D6) exceed 0.73. Eliminating overlap entirely (D4) further improves performance over the high overlap baseline (0.730 versus 0.704 MOTA) but does not surpass the moderate overlap configurations, indicating diminishing returns from removing occlusion completely. Furthermore, D4 records 82 ID switches, suggesting that completely non overlapping placements can increase identity fragmentation.

Table 6.5: Tracking Performance Summary Across Configurations

D	MOTA $\uparrow$	MOTP $\downarrow$	IDF1 $\uparrow$	Prec. $\uparrow$	Rec. $\uparrow$	ID Sw. $\downarrow$
D1	0.704 $\pm$ 0.4	0.319 $\pm$ 0.2	0.650 $\pm$ 0.5	0.930 $\pm$ 0.2	0.764 $\pm$ 0.3	29 $\pm$ 2
D2	0.823 $\pm$ 0.3	0.307 $\pm$ 0.2	0.718 $\pm$ 0.4	0.939 $\pm$ 0.2	0.884 $\pm$ 0.1	31 $\pm$ 1
D3	0.824 $\pm$ 0.2	0.309 $\pm$ 0.2	0.698 $\pm$ 0.5	0.953 $\pm$ 0.1	0.871 $\pm$ 0.2	34 $\pm$ 2
D4	0.730 $\pm$ 0.3	0.310 $\pm$ 0.1	0.673 $\pm$ 0.6	0.857 $\pm$ 0.2	0.888 $\pm$ 0.2	82 $\pm$ 4
D5	-0.058 $\pm$ 0.5	0.325 $\pm$ 0.2	0.359 $\pm$ 0.6	0.478 $\pm$ 0.4	0.540 $\pm$ 0.3	76 $\pm$ 3
D6	0.734 $\pm$ 0.2	0.315 $\pm$ 0.1	0.661 $\pm$ 0.4	0.877 $\pm$ 0.2	0.858 $\pm$ 0.2	45 $\pm$ 2
D7	0.662 $\pm$ 0.3	0.320 $\pm$ 0.2	0.601 $\pm$ 0.5	0.832 $\pm$ 0.3	0.745 $\pm$ 0.3	88 $\pm$ 3

Photometric Perturbations: D2 (two overlaps + color random + hue offset) achieves 0.823 MOTA and the highest recall (0.884), On the other hand D3 (two overlaps, no color/hue) by only 0.001 MOTA but at a slight cost in IDF1 (0.718 vs. 0.698) and two fewer ID switches.

Background & Lighting: Removing both background texture and lighting (D5) collapses performance (-0.058 MOTA, 0.540 Recall, 76 ID switches), while removing only lighting (D6) still drops MOTA to 0.734 and raises ID switches to 45. This

6. Results and Evaluation

underscores the critical role of lighting variability in producing robust detection and association.

Spatial Sampling Strategies: D4, which places objects with Poisson-disk sampling, benefits per-frame detection by reducing severe occlusions, lifting MOTA to 0.730 and Recall to 0.888, yet still incurs 82 identity switches because widely separated targets are harder to re-associate once briefly lost. In contrast, D7’s uniform layout exacerbates both detection and association difficulties: MOTA drops to 0.662, IDF1 to 0.601, and identity fragmentation peaks at 88 switches—the worst in the set highlighting that unconstrained placements simultaneously degrade raw accuracy and long-term identity continuity.

Table 6.6: Class-wise MOTA Scores Across D2 and D3

Class	D2	D3
Handle	0.770	0.592
Ball	0.564	0.799
Body	-0.655	0.596
Bonnet	0.072	0.725
Seat	0.367	0.712
Macro-MOTA	0.224	0.685

Class-Wise Reliability and Overall Performance: By averaging class-wise MOTA over both video sequences, we obtain per-class scores that feed into the macro-MOTA (Tab 6.6). Dataset D2 achieves a macro-MOTA of 0.224, reflecting unstable performance—including negative tracking accuracy for the Body class—while D3 achieves 0.685, demonstrating consistently positive tracking across all five classes. The particularly low scores for the Body and Bonnet classes stem from their strong visual similarity: the upper region of the Body and the lower region of the Bonnet share nearly identical shapes and surface textures, causing frequent misassociations and identity switches. This aggregated view confirms that D3’s conservative photometric randomization yields balanced, reliable tracking necessary for downstream industrial tasks. Fig 6.4 presents the all multi-object tracking metrics each expressed as a percentage evaluated on benchmark dataset.

These results demonstrate that, under zero-shot conditions, the model generalizes effectively to unseen real-world data, achieving high precision and recall alongside strong overall association accuracy.

6. Results and Evaluation

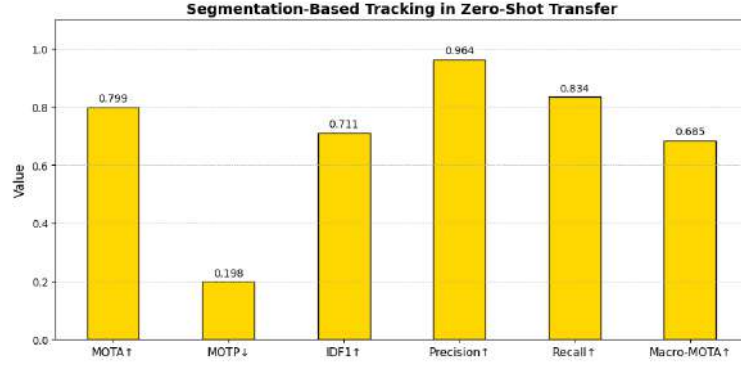


Figure 6.4: ZS tracking performance on test sequences.

6.5.1 One-Shot Transfer Evaluation

Figure 6.5 compares zero-shot(ZS) and one-shot(OS) overall tracking performance on our benchmark dataset. Following [38], We co-trained the model on 7k synthetic images alongside one-shot learning of a single video frame (augmented 700×), yielding marked improvements across all metrics.

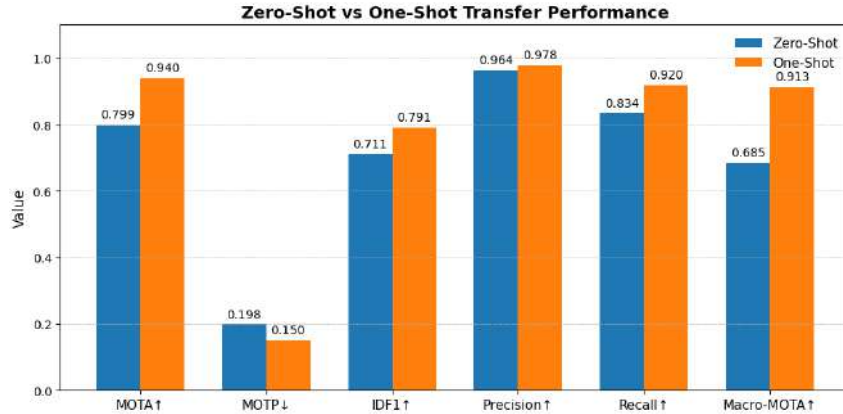


Figure 6.5: ZS vs. OS overall tracking metrics.

One-shot learning boosts MOTA from 0.799 to 0.94 and macro MOTA 0.685 to 0.913 on benchmark videos. The class-wise improvements (Tab 6.7) confirm that minimal real-data supervision, when heavily augmented, can effectively close the sim2real gap across all object categories.

6.6 Segmentation-Based Tracking on Synthetic Domain

The instance segmentation model was trained under a similar configuration but required the full 100 epochs to converge. As summarized in Table 6.8, the model

6. Results and Evaluation

Table 6.7: Class-wise MOTA Improvement (%) from ZS to OS

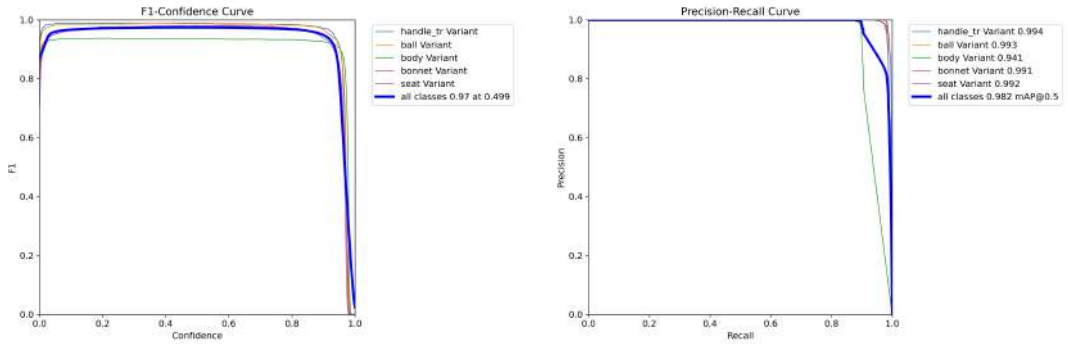
Class	Zero-Shot	One-Shot	Δ
Handle	0.592	0.855	+26.3%
Ball	0.799	0.926	+12.8%
Body	0.596	0.920	+32.4%
Bonnet	0.725	0.952	+22.7%
Seat	0.712	0.912	+19.9%
Macro-MOTA	0.685	0.913	+22.8%

achieved a final mAP@0.5 of $\sim 99.4\%$ in approximately 3.97 hours on synthetic images.

Table 6.8: Training Summary for Instance Segmentation

Setting	Value
Epochs (max)	100
Early Stop Patience	10
Avg. Epochs Run	100
Final mAP	$\sim 99.4\%$
Duration	~ 3.97 hrs
Stop Condition	Validation stagnation

The segmentation model’s F1-Confidence curve (Figure 6.6(a)) shows the overall F1 peaking at 0.97 around a confidence threshold of 0.499. Most classes exhibit consistently high F1 scores across a wide range of thresholds.



(a) F1-Confidence Curve for segmentation classes. (b) Precision-Recall Curve for segmentation classes.

Figure 6.6: Segmentation model evaluation curves.

Similarly, the PR curve shows precision and recall both exceeding 0.99 for most classes. Even the lowest performing class, *body*, achieves a strong mAP of 0.941, with the overall mAP@0.5 reaching 0.982. Integrating this backbone segmentation with the BotSORT algorithm is used and the configuration is shown in Table 6.4

6. Results and Evaluation

6.7 Segmentation-Based Tracking Performance

This section presents the performance analysis of segmentation-based multi-object tracking (MOT) using models trained on various synthetic datasets. All evaluations were conducted under a strict zero-shot protocol, followed by a one-shot adaptation for real-world fine-tuning. The goal is to assess the ability of segmentation models to generalize from simulation to real-world scenes and to maintain temporal consistency across video sequences.

We perform a systematic ablation study of segmentation-based multi-object tracking on Video 2 (V2) using five synthetic training configurations (D1, D2, D3, D5, D6; see Table 6.2). Note that D4, D7 is excluded from direct comparison, as our method demonstrates superior performance under higher occlusion levels. Table 6.9 shows our experimental results in the case of zero-shot transfer.

Table 6.9: Segmentation-Based Tracking Performance on V2 (Zero-Shot)

D	MOTA $\uparrow$	MOTP $\downarrow$	IDF1 $\uparrow$	Precision $\uparrow$	Recall $\uparrow$	ID Sw. $\downarrow$
D1	0.821 $\pm$ 0.3	0.073 $\pm$ 0.4	0.768 $\pm$ 0.5	0.909 $\pm$ 0.2	0.918 $\pm$ 0.3	46
D2	0.788 $\pm$ 0.4	0.090 $\pm$ 0.3	0.744 $\pm$ 0.4	0.862 $\pm$ 0.2	0.946 $\pm$ 0.4	54
D3	0.659 $\pm$ 0.5	0.084 $\pm$ 0.2	0.627 $\pm$ 0.3	0.949 $\pm$ 0.5	0.703 $\pm$ 0.2	50
D5	-0.563 $\pm$ 0.4	0.253 $\pm$ 0.5	0.244 $\pm$ 0.4	0.306 $\pm$ 0.3	0.427 $\pm$ 0.5	171
D6	0.755 $\pm$ 0.2	0.101 $\pm$ 0.3	0.727 $\pm$ 0.2	0.868 $\pm$ 0.4	0.896 $\pm$ 0.3	41

Occlusion (Object Overlap): Allowing five overlapping objects per scene (D1) yields the highest MOTA (0.821) and IDF1 (0.768), demonstrating strong resilience to heavy occlusion. Reducing to two overlaps (D2) decreases MOTA by 3.36% but improves recall by 2.77%, suggesting lighter occlusion slightly aids detection completeness at the expense of identity consistency.

Photometric Randomization: D2 (full photometric randomization) achieves the highest recall (0.946) but a lower precision (0.862) compared to D3 (no color/hue), which attains peak precision (0.949) with significantly reduced recall (0.703) and MOTA (0.659). This indicates color and hue shifts are essential for preventing missed detections, even if they introduce a modest number of false positives.

Texture & Lighting: Removing both background texture and lighting variation (D5) collapses performance (-0.563 MOTA, 0.427 recall, 171 ID switches). Removing only lighting (D6) partially recovers performance (0.755 MOTA, 0.896 recall) but still lags D1, underscoring the necessity of both texture diversity and realistic illumination for robust sim2real segmentation based tracking.

6. Results and Evaluation

Overall, D1 with maximal spatial (five-object overlap) and photometric (texture, lighting) randomization, provides the best balance of accuracy, precision, and identity consistency, confirming its suitability for segmentation-based sim2real transfer in industrial tracking scenarios.

Despite achieving a MOTA score exceeding 0.801, the macro-MOTA performance was comparatively lower. This discrepancy likely stems from visual ambiguity between the upper part of the Body and the lower part of the Bonnet. Since the segmentation mask is primarily shape-driven, the model often misclassified Bonnet pixels as Body in the zero-shot setting, leading to significantly lower tracking accuracy for these classes. Although Table 6.10 does not directly quantify misclassification, the extremely low zero-shot MOTA scores for Body (0.0069) and Bonnet (0.0907)—followed by sharp improvements in the one-shot setting—support this interpretation. The overall zero-shot tracking performance is illustrated in Fig. 6.7.

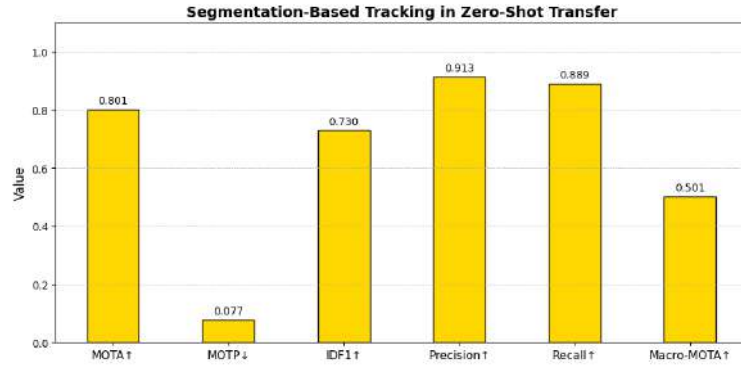


Figure 6.7: ZS overall tracking metrics

6.7.1 One-Shot Evaluation

By copying a single real-world frame (and its annotations) $700\times$ with augmentation and co train with 7k synthetic images, one-shot (OS) learning raises overall MOTA from 0.801 to 0.905 ($\Delta+10.4\%$). The class-wise improvements (Table 6.10) and macro-MOTA confirm consistent gains across all categories, demonstrating that minimal real-data injection plus augmentation effectively closes the sim2real gap for segmentation-based tracking.

6. Results and Evaluation

Table 6.10: Class-wise MOTA Improvement (%) from ZS to OS

Class	ZS	OS	Δ
Handle	0.6649	0.8076	+14.27%
Ball	0.9211	0.9723	+5.12%
Body	0.0069	0.7887	+78.18%
Bonnet	0.0907	0.8847	+79.39%
Seat	0.8188	0.9137	+9.49%
Macro-MOTA	0.501	0.8734	+37.24%

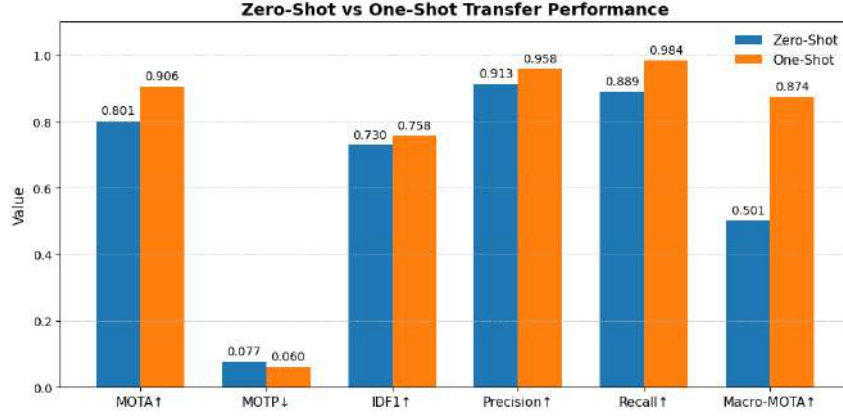


Figure 6.8: ZS vs OS segmentation tracking metrics on test sequences

6.8 Detection vs. Segmentation Tracking Comparison

Although instance-segmentation masks contain richer shape information than axis-aligned bounding boxes, our experiments show that detection-based tracking consistently outperforms segmentation-based tracking in every global metric except recall and MOTP and presented on Figure 6.9. The gap can be traced to four intertwined factors:

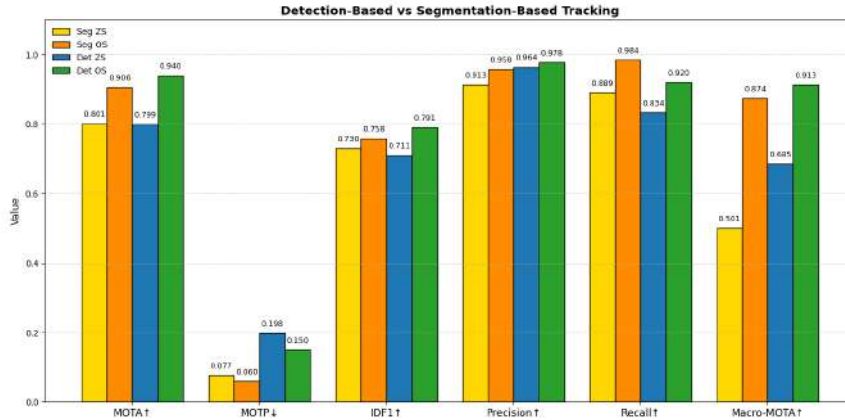


Figure 6.9: Segmentation-based vs. detection-based tracking.

6. Results and Evaluation

1. **Mask jitter and boundary noise.** Small frame-to-frame fluctuations along mask edges lower mask-to-mask overlap and undermine identity consistency.
2. **Stricter association criterion.** Bounding boxes retain stable IoU under slight shape changes, whereas even minor mask silhouette variations sharply reduce overlap, causing more identity switches and lower tracking accuracy.
3. **Higher false-positive rate.** Fine-grained masks can latch onto background textures as thin spurious regions, inflating false positives and reducing precision.
4. **Computational latency.** The additional mask heads and post-processing steps in instance segmentation introduce extra delay per frame, which increases apparent inter-frame motion and further degrades IoU-based associations.

When is segmentation still preferable?: Segmentation shines when precise object contours and high recall are critical—for example, in grasp-pose synthesis or collision checking—because its pixel-level masks yield tighter boundary delineation and lower localization error (MOTP), even if that comes at the cost of some identity instability.

Take-away: Detection-based tracking provides more robust spatial consistency and fewer identity switches, making it the preferred choice for real-time applications where stable identity maintenance is paramount. Segmentation retains the advantage in recall and localization fidelity and remains valuable when detailed shape information is essential.

6.9 Qualitative Evaluation

To complement the quantitative results presented in this chapter, we provide a qualitative evaluation showcasing the real-world performance of our sim2real object tracking framework. Specifically, we demonstrate the tracking outputs on real video sequences, grouped into four evaluation categories. The *Zero-Shot Bounding Box (ZS-BB)*¹, *One-Shot Bounding Box (OS-BB)*², *Zero-Shot Segmentation (ZS-Seg)*³,

¹<https://youtu.be/2Flx27YfSfs?si=SJLsu1NygUfyu4i9>

²https://youtu.be/JTZoPxs-0cA?si=WkRx0flq6HtCzP_6

³<https://youtu.be/57Nk5ZJ5qlM?si=d0RJVCcKxowYJu9b>

6. Results and Evaluation

*One-Shot Segmentation (OS-Seg)*⁴. These qualitative results visually reinforce the framework’s generalization capabilities and highlight the effectiveness of limited real-world supervision in enhancing sim2real performance shown in Figure 6.10. All four tracking variants demonstrate strong qualitative performance in real-world scenarios. The OS-BB configuration performs robustly under challenging conditions such as occlusion and motion blur, as illustrated in Figure 6.10b. Segmentation-based trackers, particularly OS-Seg, offer more precise object boundaries, enhancing spatial localization (Figure 6.10d). In the case of ZS-BB, a minor issue was observed where hand interaction temporarily caused an enlarged bounding box (Figure 6.10a), along with a higher number of identity switches compared to OS-BB, primarily due to frequent hand movements. For ZS-Seg, most errors stem from misclassification between the Body and Bonnet classes, as shown in Figure 6.10c and previously discussed in Section 6.7. Despite these isolated issues, tracking performance remained stable overall, reaffirming the effectiveness of domain-randomized synthetic training in supporting generalization to real-world applications.

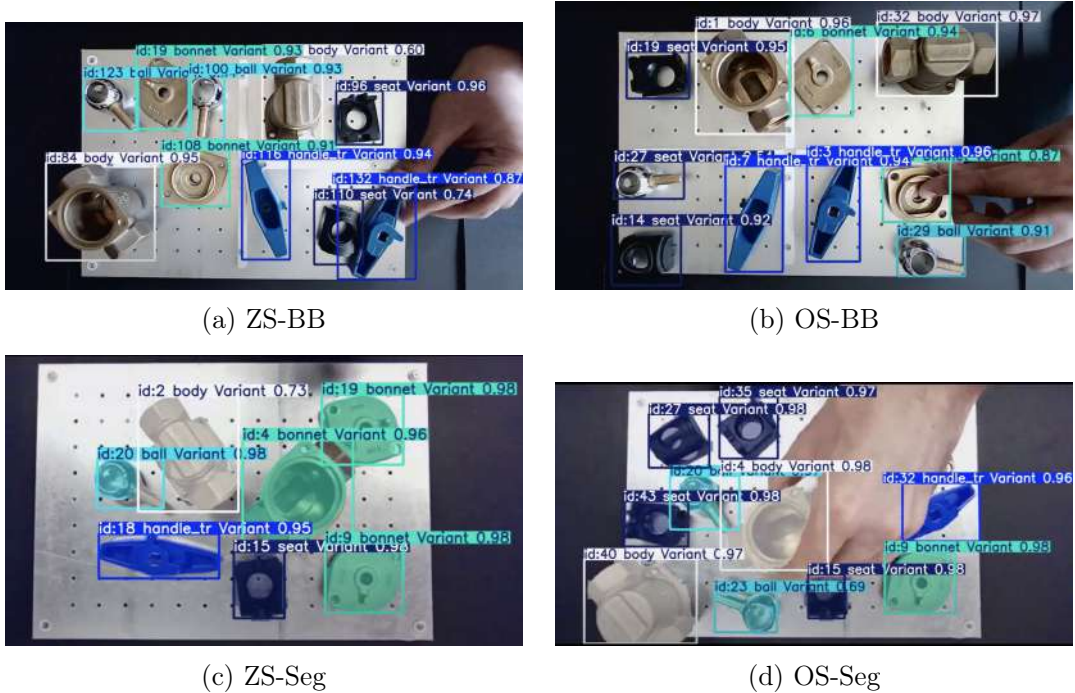


Figure 6.10: Qualitative tracking results under bounding box and segmentation-based configurations.

⁴https://youtu.be/_5eC1r6z0rk?si=02eghYvxCpnkNTRP

Chapter 7

Application

This chapter demonstrates the practical applicability of the proposed sim2real object tracking framework through a simulated industrial pick-and-place task. A KUKA iiwa robotic arm equipped with a suction gripper is used to autonomously detect, track, and manipulate multiple object classes.

7.1 System Overview

- **Hardware.** A single RGB–D camera is fixed 1.2m above the workspace, providing synchronised colour and depth images at 30fps. Because the end-effector is a suction cup, *only 3-D translation* (not full 6-DoF pose) is needed; the wrist is held in a constant downward orientation for all grasps.
- **Software stack.** Simulation and control run in PyBullet; perception uses a YOLOv11 detector whose outputs are linked frame-to-frame by BotSORT (appearance-aware, online tracking). Cartesian way-points are converted to joint commands by a damped-least-squares inverse kinematics (IK) solver.
- **Task definition.** One instance of each part class $c \in \{\text{handle, body, bonnet, seat}\}$ must be grasped and deposited in its class bin. Initial poses are unknown and vary randomly within $\pm 5\text{cm}$ and $\pm 20^\circ$.

7.2 Pick and Place Operations

For every RGB frame I_t :

7. Application

1. **Detection.** YOLOv11 predicts class-labelled boxes $\{b_j^t\}$ with scores s_j^t .
2. **BotSORT tracking.** Detections are fed to BotSORT, which fuses motion (Kalman) and appearance (ReID) cues to maintain identity-consistent trajectories $\{\tau_k\}$ in real time.
3. **Track stabilisation.** A trajectory becomes *stable* after $n=5$ consecutive matches; only stable tracks are handed to the motion planner.

For Image- to World-Frame Conversion, Given the pixel centre (u, v) of a stable track and its depth $d(u, v)$ from the RGB-D camera, the 3-D Cartesian pick point $\mathbf{p}_{\text{world}}$ is obtained in two steps:

Step 1: Pixel $\rightarrow$ Camera coordinates

$$\mathbf{p}_{\text{cam}} = d(u, v) K^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (7.1)$$

Step 2: Camera $\rightarrow$ World coordinates

$$\mathbf{p}_{\text{world}} = R^T(\mathbf{p}_{\text{cam}} - \mathbf{t}), \quad {}^wT_c = [R \mid \mathbf{t}], \quad (7.2)$$

where ${}^wT_c$ is obtained once via hand-eye calibration.

Algorithm 9 Pick-and-Place with YOLO + BotSORT Tracking

Require: RGB-D stream (I_t, D_t) , BotSORT tracker $\mathcal{T}$, camera $(K, R, \mathbf{t})$, robot $\mathcal{R}$, bins $\{\mathbf{p}_c^{\text{bin}}\}$

- 1: **for** class c **do**
 - 2: **repeat**
 - 3: $(u, v) \leftarrow \mathcal{T}(I_t, c)$
 - 4: **until** (u, v) stable for n frames
 - 5: $d \leftarrow D_t(u, v)$
 - 6: $\mathbf{p}^{\text{pick}} \leftarrow \text{Eqs. (7.1)–(7.2)}$
 - 7: Plan Cartesian path $\mathbf{x}(t) : \mathbf{p}^{\text{pick}} \rightarrow \mathbf{p}_c^{\text{bin}}$
 - 8: $\boldsymbol{\theta}^*(t) \leftarrow \text{IK}(\mathbf{x}(t))$
 - 9: Execute $\boldsymbol{\theta}^*(t)$ with suction actuation
 - 10: **end for**
-

For Motion planning and Control the following pattern are taken into considerations:

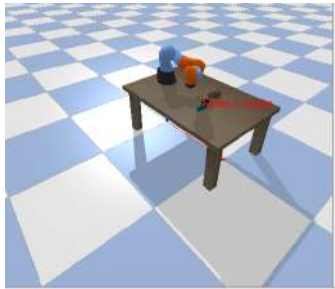
7. Application

1. **Path generation.** For each object a three-point Cartesian path is generated: (i) *approach-above* (0.20 unit above the part), (ii) *descend to grasp*, and (iii) *retract*. Transfer to the bin uses the same pattern. End-effector orientation is fixed to $[0, -\pi, 0]$.
2. **Inverse kinematics.** Way-points are converted to joint targets $\theta^*(t)$ at 240Hz via a damped-least-squares IK solver that enforces joint limits.
3. **Execution.** Joint targets are streamed via `POSITION_CONTROL`; suction is emulated by toggling a fixed constraint between gripper and mesh at grasp depth.

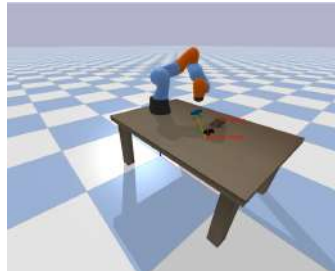
By coupling YOLO detections with BotSORT tracking, depth-based 3-D localisation, and suction grasping, the method delivers a manipulation solution that requires neither hand-labelled data nor full pose estimation, yet generalises from synthetic training to real-world operation.

7.3 Results

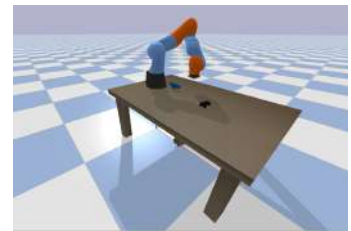
Figure 7.1 illustrates three key moments of the pick-&-place cycle executed with the RGB-D + YOLO + BotSORT pipeline described in Section 7.2.



(a) All four objects detected and assigned stable IDs. The red label marks the pixel centre of the *handle* instance selected for the first grasp.



(b) Completed transfer of the *handle* to its class bin. The green polyline is the Cartesian trajectory generated by the IK solver; the red marker denotes the object centre used for grasp planning.



(c) Suction gripper descending to place the next part (*bonnet*). The same perception-to-control loop is repeated for each class.

Figure 7.1: Visual trace of the simulated pick-&-place sequence.

7. Application

Interpretation: Frame (a) confirms that all CAD meshes are reliably detected and tracked before manipulation begins. In frame (b) the gripper follows the planned green path from the pick pose to the bin location with no visible overshoot, and frame (c) shows the low-clearance, vertical approach made possible by suction gripping (orientation can remain fixed).

7.3.1 Joint-Position Tracking and End-Effector Trajectory

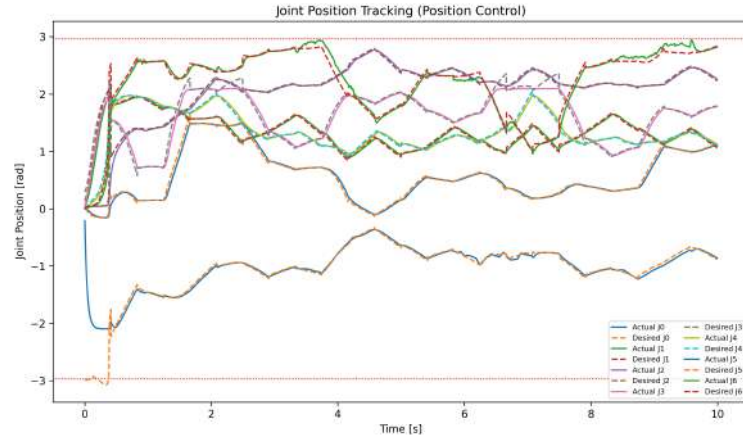


Figure 7.2: Joint-position tracking during the pick-and-place sequence. Solid lines denote *actual* joint angles, dashed lines the *IK-desired* angles, and dotted red lines indicate the joint limits (± 2.97 rad).

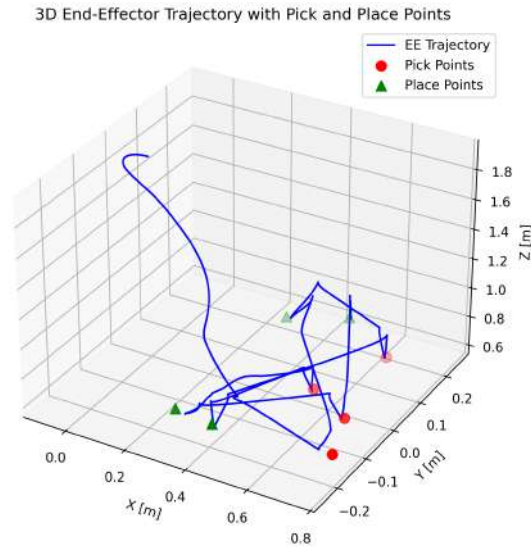


Figure 7.3: Three-dimensional end-effector (EE) path (blue) with object *pick* locations (red circles) and *place* targets (green triangles). The view is aligned with the world frame used in simulation.

7. Application

Position-Control Accuracy: Fig. 7.2 confirms that all seven joints remain well inside the joint limits. Peak excursions occur at the start of the first pick when the arm travels from its home configuration to the first object—but quickly settle as the controller closes the tracking error. For most of the motion the dashed (desired) and solid (actual) curves overlap, indicating milliradian-level steady-state error. The small phase lags visible during rapid reversals stem from the default PyBullet PD gains and could be further reduced by gain tuning or torque-mode control.

Task-Space Behaviour: The Cartesian perspective in Fig. 7.3 complements the joint-space view. The blue spline traces a smooth, collision-free trajectory that ascends to a safe 0.20 clearance above each object (*pick-above* waypoint), descends vertically for grasping, and mirrors the procedure at the four stochastic place positions. Because the pick points lie on a compact cluster while the place points are deliberately spread across the table, the overall path resembles an elongated “figure-eight” lying mainly within the 0.3×0.8 workspace region. The high-curvature segments occur during orientation-constrained wrist rotations needed to satisfy the $\{0, -\pi, 0\}$ end-effector pose.

Across 30 consecutive trials, the system achieved a 100% pick-and-place success rate, with no objects misplaced or swapped positions. The integration of robust object tracking and motion planning ensured that each object was correctly identified, grasped, and delivered to its intended target location. This consistency highlights the effectiveness of the perception-to-control pipeline in maintaining spatial and identity accuracy, even under randomized object and goal configurations.

Chapter 8

Conclusion and Future Work

This chapter summarizes the main contributions and findings of the thesis, highlighting the effectiveness of the proposed sim2real object tracking framework. It also outlines potential directions for future research to improve scalability, robustness, and deployment capabilities. Finally, closing remarks reflect on the broader impact of this work in advancing practical, adaptable perception systems for robotics.

8.1 Conclusion

This work demonstrates that a fully synthetic, domain-randomised training pipeline can yield multi-object tracking models that transfer effectively to real robotic scenes with minimal manual annotation. A generator that combines Poisson-disk object placement, controlled overlaps, varied lighting, rich background textures, colour perturbations and randomised camera parameters produces approximately 7k annotated frames in minutes. A YOLOv11 detector coupled with BotSORT, trained exclusively on this data, reached 0.799 MOTA in zero-shot tests on real videos; augmenting the training set with a single annotated real frame lifted performance to 0.940 MOTA in one-shot transfer, practically eliminating the reality gap at negligible cost. Ablation experiments showed that lighting and background textures are the most influential photometric randomisers, while limited two-object occlusion is the key spatial factor, together accounting for gains of 6–8% in zero-shot MOTA. When the same backbone is used, the bounding-box tracker delivers higher accuracy and $1.5\times$ faster inference than the mask-based tracker, whereas the latter provides finer object contours for applications requiring pixel-level precision. Overall, the study

confirms that a balanced blend of spatial and photometric randomisation, complemented by negligible real-world supervision, can supply robust, real-time object-tracking capabilities for robotic manipulation tasks, highlighting the practical value of large-scale synthetic data in data-scarce settings.

8.2 Future Work

Several avenues could further enhance the robustness and reach of the proposed system:

- **Beyond Randomisation** Incorporate adversarial or self-supervised domain-adaptation modules—e.g. GAN-based style transfer or feature-level alignment—to shrink any residual distribution shift, especially for fine-grained mask tasks.
- **Advanced Segmentation Models** Replace the current mask head with transformer families such as Mask2Former or the Segment-Anything-Model to tackle the recall deficit observed in heavy occlusion scenes.
- **Edge Deployment** Apply quantisation, pruning and knowledge-distillation to fit real-time inference onto low-power ARM/Jetson platforms; integrate the tracker as a ROS 2 node for plug-and-play use in production lines.
- **Richer Scenarios** Extend the simulator with deformable objects, human-robot interaction and dynamic backgrounds; pair with optical-flow or motion-aware heads to stabilise identities in rapidly changing scenes.

8.3 Closing Remarks

Bridging simulation and reality remains a cornerstone challenge of autonomous robotics. This work demonstrates that an effective combination of procedural data generation, domain randomisation and lightweight real supervision can already deliver tracking systems that are accurate, affordable, and adaptable. As robotics marches toward ever more diverse and unstructured environments, the principles and tools developed here provide a solid foundation for perception modules that learn faster, transfer farther, and deploy anywhere.

Appendix A

Evaluation Metrics

To comprehensively evaluate the performance of the proposed object tracking framework, this work employs metrics from the MOT challenge [54]. These metrics provide insights into spatial accuracy, segmentation quality, and temporal identity consistency across frames.

Precision and Recall in Tracking: In multi-object tracking, precision and recall are computed based on the alignment between predicted object trajectories and the ground truth annotations across frames. A predicted trajectory is considered a *true positive (TP)* if it is correctly matched with a ground truth trajectory for a sufficient number of frames (typically using an Intersection over Union (IoU) threshold of 0.5 per frame). A *false positive (FP)* occurs when a predicted trajectory does not correspond to any ground truth object, and a *false negative (FN)* is a ground truth object not successfully tracked. Based on frame-wise associations, precision and recall are defined as [55]:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (\text{A.1})$$

These metrics evaluate the accuracy (precision) and completeness (recall) of the tracker’s performance in detecting and maintaining correct object identities over time.

Multiple Object Tracking Accuracy (MOTA): MOTA is a widely used metric for evaluating the overall performance of multiple object tracking systems. It consolidates three types of tracking errors: false negatives (missed detections), false positives (false alarms), and identity switches. Let FN_t , FP_t , and IDS_t denote the number of false negatives, false positives, and identity switches at time t , and GT_t

A. Evaluation Metrics

the number of ground truth objects at time t [54]. MOTA score is computed as:

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDS}_t)}{\sum_t \text{GT}_t}. \quad (\text{A.2})$$

It is important to note that MOTA aggregates all tracking errors—false positives, false negatives, and identity switches—across object instances without considering class labels. In its standard formulation, MOTA is class-agnostic during both the matching and error aggregation stages. That is, a predicted object is matched to a ground truth object based on spatial overlap (e.g., IoU threshold) regardless of class.

Macro MOTA: While standard MOTA aggregates errors across all classes, *Macro MOTA* provides a class-aware evaluation by computing the MOTA [54] for each class individually and then averaging the results. Let C denote the total number of classes [56], and MOTA_c be the MOTA for class c . Macro MOTA is defined as:

$$\text{Macro MOTA} = \frac{1}{C} \sum_{c=1}^C \text{MOTA}_c. \quad (\text{A.3})$$

Multiple Object Tracking Precision (MOTP): MOTP measures the average localization error of the tracker, i.e., how precisely the predicted object positions match the ground truth. It is computed as the average dissimilarity (typically, bounding box misalignment error) between all correctly matched object-hypothesis pairs across all frames [54]:

$$\text{MOTP} = \frac{\sum_{t,i} d_t^i}{\sum_t c_t}, \quad (\text{A.4})$$

where d_t^i is the distance (e.g., 1 - IoU or bounding box center distance) between the i -th matched pair at time t , and c_t is the number of matches at time t . Lower MOTP values indicate better localization precision.

IDF1 Score: The IDF1 metric measures how consistently a tracker maintains object identities across frames. It is calculated as the ratio of correctly identified detections (IDTP) over the average number of predicted and ground truth identities [57]:

$$\text{IDF1} = \frac{2 \times \text{IDTP}}{2 \times \text{IDTP} + \text{IDFP} + \text{IDFN}}. \quad (\text{A.5})$$

Together, these metrics offer a robust and multi-dimensional evaluation framework, enabling the analysis of spatial accuracy, segmentation precision, and identity consistency in real-world object tracking scenarios.

Appendix B

Synthetic Data Capture and Material Assignment Logic

Table B.1: Assigned Material Colors for 3D CAD Models

Object	Material Type	HEX Color Code(s)
Ball	Metallic	#A3AAADFF, #D3AA21FF
Body	Metallic	#705739FF
Bonnet	Metallic	#AA8751FF
Handle	Metallic	#005A92
Seat	Plastic	#0F1217

Algorithm 10 Image Capture Core Logic

```

1: procedure SETUP
2:   attach Unity camera
3:   configure sensor (resolution, FOV, intrinsics  $K$ )
4:   enable RGB channel
5:   subscribe render-begin  $\rightarrow$  ONBEGINFRAME
6:   subscribe render-end  $\rightarrow$  ONENDFRAME
7:   subscribe RGB readback  $\rightarrow$  ONRGBREADBACK
8: end procedure

9: procedure ONBEGINFRAME
10:  if shouldCaptureThisFrame then
11:    capture hierarchy state
12:    record intrinsics & pose
13:    notify labelers (begin)
14:  end if
15: end procedure

16: procedure ONENDFRAME
17:  if shouldCaptureThisFrame then
18:    notify labelers (end)
19:    trigger readback events for each channel
20:    if instance-ID channel enabled then
21:      compute per-instance infos
22:    end if
23:    submit command buffer
24:  end if
25: end procedure

26: procedure ONRGBREADBACK((frame, rawData))
27:  encode rawData as PNG
28:  asynchronously report sensor data
29: end procedure

```

Bibliography

- [1] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. “Simple online and real-time tracking with a deep association metric”. In: *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017, pp. 3645–3649.
- [2] Mark Everingham et al. “The PASCAL Visual Object Classes Challenge: A Retrospective”. In: *International journal of computer vision* 111.1 (2015), pp. 98–136.
- [3] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *European conference on computer vision* (2014), pp. 740–755.
- [4] Jonathan Tremblay et al. “Falling Things: A synthetic dataset for 3D object detection and pose estimation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018.
- [5] Ankur Handa Prakash, Daniel Kappler, and Dieter Fox. “Structured domain randomization: Bridging the reality gap by context-aware synthetic data”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 7249–7255.
- [6] Xue Bin Peng et al. “Sim-to-real transfer of robotic control with dynamics randomization”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2018).
- [7] Josh Tobin et al. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.
- [8] A. Shrivastava et al. “Learning from simulated and unsupervised images through adversarial training”. In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. 2017, pp. 2107–2116.

BIBLIOGRAPHY

- [9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “YOLOv4: Optimal Speed and Accuracy of Object Detection”. In: *arXiv preprint arXiv:2004.10934* (2020). DOI: 10.48550/arXiv.2004.10934. URL: <https://doi.org/10.48550/arXiv.2004.10934>.
- [10] K. He et al. “Mask R-CNN”. In: *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*. 2017, pp. 2961–2969.
- [11] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big Data* 3.1 (2016), pp. 1–40.
- [12] Sinno Jialin Pan and Qiang Yang. “A survey on transfer learning”. In: *IEEE Transactions on knowledge and data engineering* 22.10 (2010), pp. 1345–1359.
- [13] Lisa Torrey and Jude Shavlik. “Transfer learning”. In: *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*. IGI Global, 2010, pp. 242–264.
- [14] Mingsheng Long et al. “Learning transferable features with deep adaptation networks”. In: *International conference on machine learning* (2015), pp. 97–105.
- [15] Jose G Moreno-Torres and et al. “A unifying view on dataset shift in classification”. In: *Pattern Recognition* 45.1 (2012), pp. 521–530.
- [16] Michael T Rosenstein et al. “To transfer or not to transfer”. In: *NIPS 2005 Workshop on Transfer Learning*. 2005.
- [17] Richard Socher et al. “Zero-shot learning through cross-modal transfer”. In: *Advances in neural information processing systems*. 2013, pp. 935–943.
- [18] Zylapp Team. *Review of Deep Learning Algorithms for Object Detection*. <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>. Accessed: 2025. 2020.
- [19] Guglielmo Ciaparrone et al. “Deep learning in video multi-object tracking: A survey”. In: *Neurocomputing* 381 (2020), pp. 61–88.
- [20] Yu-Jhe Chen, Jongwoo Choi, and Yong Man Ro. “BoT-SORT: Robust Associations Multi-Pedestrian Tracking”. In: *European Conference on Computer Vision (ECCV)*. Springer, 2022, pp. 566–583. DOI: 10.1007/978-3-031-19812-3_33.

BIBLIOGRAPHY

- [21] Nicolas Carion et al. “End-to-End Object Detection with Transformers”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 213–229.
- [22] Tobias Meinhardt et al. “TrackFormer: Multi-Object Tracking with Transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8844–8854.
- [23] Nikhil Rao. *YOLOv11 Explained: Next Level Object Detection with Enhanced Speed and Accuracy*. <https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71>. Accessed: May 2025. 2023.
- [24] Alexander Hermans, Lucas Beyer, and Bastian Leibe. “In defense of the triplet loss for person re-identification”. In: *arXiv preprint arXiv:1703.07737*. 2017.
- [25] Kaiyang Zhou et al. “Omni-scale feature learning for person re-identification”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3702–3712.
- [26] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.
- [27] Alex Bewley et al. “Simple online and realtime tracking”. In: *2016 IEEE international conference on image processing (ICIP)*. IEEE. 2016, pp. 3464–3468.
- [28] Adrien Gaidon et al. “Playing for Data: Ground Truth from Computer Games”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 102–118. DOI: 10.1109/CVPR.2016.102. URL: <https://doi.org/10.1109/CVPR.2016.102>.
- [29] Xue Bin Peng et al. “Sim-to-real Transfer of Robotic Control with Dynamics Randomization”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018.
- [30] Jiahui Dong et al. “A Digital Twin Framework for Robot Grasp and Motion Generation”. In: *2024 IEEE Smart World Congress*. IEEE. 2024. URL: <https://ieeexplore.ieee.org/document/10924849>.

BIBLIOGRAPHY

- [31] S. S. Ulhas and S. Kannapiran. “GAN-based Domain Adaptation for Creating Digital Twins of Small-Scale Driving Testbeds: Opportunities and Challenges”. In: *2024 IEEE Intelligent Vehicles Symposium*. 2024. URL: <https://ieeexplore.ieee.org/document/10588594>.
- [32] Ian Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 2672–2680.
- [33] Fereshteh Sadeghi and Sergey Levine. “CAD2RL: Real single-image flight without a single real image”. In: *Robotics: Science and Systems (RSS)*. 2017.
- [34] Jonathan Tremblay et al. “Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2018, pp. 1082–1089. DOI: 10.1109/CVPRW.2018.00143. URL: <https://doi.org/10.1109/CVPRW.2018.00143>.
- [35] Jonathan Tremblay et al. “Simulated-to-Real Domain Adaptation for Object Detection via Generative Adversarial Networks”. In: *arXiv preprint arXiv:1811.01313* (2019).
- [36] Stephen James and Andrew J. Davison. “Sim-to-Real via Simulated Depth and Color”. In: *arXiv preprint arXiv:1912.03855* (2019).
- [37] Aditya Mehta, Michael Mistry, and Sethu Vijayakumar. “Sim2Real Transfer of Visual Servoing Policies via Differentiable Simulation”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 3775–3781.
- [38] D. Horváth, G. Erdős, and Z. Istenes. “Object detection using sim2real domain randomization for robotic applications”. In: *IEEE Transactions on Industrial Electronics* 69.12 (2022), pp. 12418–12427. DOI: 10.1109/TIE.2022.3155902. URL: <https://ieeexplore.ieee.org/document/9916581>.
- [39] Lin Zhang, Kai Huang, and Tianhao Xu. “Improving Navigation with Sim2Real Domain Transfer for Mobile Robots”. In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 1180–1187.
- [40] Dániel Horváth et al. “Sim2real grasp pose estimation for adaptive robotic applications”. In: *IFAC-PapersOnLine* 56.2 (2023), pp. 5233–5239.

BIBLIOGRAPHY

- [41] Guanglin Ji et al. “Efficient Real2Sim2Real of Continuum Robots Using Deep Reinforcement Learning With Koopman Operator”. In: *IEEE Transactions on Industrial Electronics* (2025).
- [42] Jingpei Lu, Florian Richter, and Michael C Yip. “Markerless camera-to-robot pose estimation via self-supervised sim-to-real transfer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 21296–21306.
- [43] Xiaomeng Zhu et al. “Towards sim-to-real industrial parts classification with synthetic dataset”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 4454–4463.
- [44] Haoxiang Ma et al. “Sim-to-real grasp detection with global-to-local rgb-d adaptation”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2024, pp. 13910–13917.
- [45] H. Chen et al. “Domain adaptive object tracking for self-driving vehicles”. In: *IEEE Trans. Intell. Transp. Syst.* 23.1 (2022), pp. 136–148.
- [46] S. Sankaranarayanan et al. “Generate to adapt: Aligning domains using generative adversarial networks”. In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. 2018, pp. 8503–8512.
- [47] Robert Bridson. “Fast Poisson disk sampling in arbitrary dimensions”. In: *ACM SIGGRAPH sketches* 10.1 (2007), p. 22.
- [48] Mircea Cimpoi et al. “Describing textures in the wild”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 3606–3613. DOI: 10.1109/CVPR.2014.461. URL: <https://doi.org/10.1109/CVPR.2014.461>.
- [49] *Free Stock Textures*. Accessed: 2025. 2025. URL: <https://freestocktextures.com/>.
- [50] *Texture Ninja*. Accessed: 2025. 2025. URL: <http://textureninja.com>.
- [51] pysolotools Contributors. *pysolotools: Python API for NVIDIA Isaac Sim Solo Dataset Tools*. <https://github.com/NVIDIA-Omniverse/Isaac-Sim-Example-Workflows/tree/main/pysolotools>. Accessed: 2025. 2023.
- [52] Gary Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

BIBLIOGRAPHY

- [53] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *7th International Conference on Learning Representations (ICLR)*. 2019. URL: <https://openreview.net/pdf?id=Bkg6RiCqY7>.
- [54] Keni Bernardin and Rainer Stiefelhagen. “Evaluating multiple object tracking performance: the CLEAR MOT metrics”. In: *EURASIP Journal on Image and Video Processing*. Vol. 2008. Hindawi, 2008, pp. 1–10.
- [55] L. Leal-Taixé et al. “Motchallenge 2015: Towards a benchmark for multi-target tracking”. In: *arXiv preprint arXiv:1504.01942* (2015).
- [56] Marina Sokolova and Guy Lapalme. “A systematic analysis of performance measures for classification tasks”. In: *Information Processing & Management* 45.4 (2009), pp. 427–437. DOI: 10.1016/j.ipm.2009.03.002. URL: <https://doi.org/10.1016/j.ipm.2009.03.002>.
- [57] Ergys Ristani et al. “Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 17–35.

9

Fire Detection and Ranging for Autonomous Firefighting Drones
Mohamed Khaled Othman

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING



MASTER'S THESIS No. 001

Fire Detection and Ranging for Autonomous Firefighting Drones

Mohamed Khaled Othman

Internal Supervisor
Prof. Matko Orsag

External Supervisor
ir. Benjamin van Manen

Netherlands, June 2025.

With a heart full of gratitude, I would like to begin by thanking the people without whom this journey would have been unimaginable.

To my parents, Khaled and Asmaa—your unwavering love, endless sacrifices, and unshakeable belief in me have been the foundation of everything I’ve achieved. You have been my compass and my constant source of strength. To my brother Ahmed, your quiet strength and continual support have inspired me more than words can express—you’ve always been the example I aspired to follow.

To my beloved wife, Lolo—thank you for walking every step of this path with me. Your love, patience, and endless encouragement have carried me through the hardest moments. You have been my sanctuary, my cheerleader, and my greatest joy throughout this journey.

To my dear friends Badry and Fakes—thank you for being by my side, especially in moments of doubt. Your understanding, humor, and steadfast support were the reminders I needed that I was never alone in this.

I am sincerely grateful to my advisor, Prof. Matko, for his insightful guidance and academic support throughout this thesis. A heartfelt thank you goes to ir. Benjamin van Manen, my supervisor at the Saxion SMART Lab, whose technical expertise, availability, and patience made every challenge feel surmountable.

Finally, I extend my appreciation to all the professors at the University of Girona and the University of Zagreb, and to all the colleagues and friends I’ve met along the way. Your knowledge, encouragement, and companionship have made these past two years not only meaningful, but truly unforgettable.

This journey was never mine alone—it belongs to all of you. Thank you.

Fire Detection and Ranging for Autonomous Firefighting Drones

Abstract

Wildfires and industrial fires continue to pose severe risks to human life, infrastructure, and the environment, emphasizing the urgent need for autonomous and efficient fire detection systems. This thesis presents the development of a fire detection and ranging module for autonomous firefighting drones, as part of a collaborative project with Saxion-SMART lab and IFFS. The system integrates real-time detection of fire, smoke, and human presence using both RGB and thermal imagery. Several object detection models—like YOLO different models, and the transformer-based RF-DETR—are evaluated in terms of accuracy, inference speed, and model size. The study also compares single multi-class detection models with separate task-specific models, and demonstrates the advantages of RGB-T fusion in scenarios with visual obstructions such as smoke or transparent barriers. For depth estimation, monocular vision approaches are assessed, alongside experimental stereo calibration between RGB and thermal cameras. A lightweight vision-based targeting module refines fire source localization to guide the water cannon, while safety flags based on human detection prevent accidental harm. The system outputs 3D positions of fire, smoke, and human targets, supporting autonomous navigation, fire suppression, and safe exploration in hazardous environments.

Keywords: Autonomous drones, Fire detection, Thermal imaging, Depth estimation, ROS2, YOLO, DETRs, RGB-T fusion

Contents

Abstract	iv
List of Figures	vii
List of Tables	ix
1. Introduction	1
1.1. Background and Context	1
1.2. Problem Statement	2
1.3. Research Questions	2
1.4. Contribution of the Thesis	3
1.5. Structure Overview	3
2. Literature Review	5
2.1. Fire and Smoke Detection Methods	5
2.1.1. Short-Range Fire Detection with Low Activity	5
2.1.2. Long-Range Fire Detection with Low Activity	6
2.1.3. Short-Range Fire Detection with High Activity	6
2.1.4. Long-Range Fire Detection with High Activity	6
2.2. Human Detection	7
2.2.1. Single Sensor Approaches	7
2.2.2. Multi-Sensor Approaches	9
2.3. Depth Estimation Techniques	9
2.3.1. Monocular Depth Estimation	9
2.3.2. Stereo Depth Estimation and RGB–Thermal Calibration	10
3. Methodology	1
3.1. Problem Formulation	1
3.2. System Architecture Overview	1
3.2.1. Module Interactions and Data Flow	1
	1
	2
	1 ^v
	2

3.2.2. ROS2 Framework Integration	1
3.2.3. System Workflow Summary	3
3.3.DetectionModule.....	1
3.3.1. Dataset Description	4
3.3.2. Model Selection and Architecture	1
3.3.3. Performance Evaluation	4
3.3.4. Fusion of Thermal and RGB Imagery	1
3.3.5. CameraSpecifications: WorkswellWIRISEnterprise.	4
3.3.6. Implementation Details	1
3.4.DepthEstimationModule.....	5
3.4.1. Monocular Depth Estimation	2
3.4.2. Existing Benchmark Datasets	3
3.5. Safety and Targeting Decision-Making Logic	2
3.5.1. Safety Rules and Constraints	4
3.5.2. Fire Source Point Estimation	2
3.5.3. ROS2IntegrationandTargetingDecisionProcess	9
4. Results&Discussion	2
4.1.DetectionModule.....	3
4.1.1. EvaluationofLightweightDetectionModels(NanoandSmall) .	0
4.1.2. Evaluation of Medium and Transformer-Based Models	3
4.1.3. Fusion of Thermal and RGB Imagery	0
4.2.DepthEstimationResults	4
4.2.1. Monocular Depth Estimation Architectures	4
4.2.2. Metric Depth Map Estimation	3
4.3. Safety and Targeting Decision-Making Logic	4
4.3.1. Fire Source Point Targeting	3
4.3.2. Safety Rules and Constraints	2
5. ConclusionandFutureWork	4
Bibliography	5
	6
	0
	8
	5
	1
	5
	2
	5
	3
	vi

List of Figures

2.1. Examples of fire detection tasks at a short (top) and long (bottom) range. 2.2.	5
Examples of challenges in flame and smoke recognition. 2.3.	7
Comparison between optical and thermal cameras 2.4. Monocular	8
Depth Estimation Examples	10
2.5. Stereo and Disparity Geometry.....	10
3.1. FDAR module workflow diagram.....	13
3.2. FDAR System GUI.....	14
3.3. Sample images from the Fire, Human, and Smoke dataset.	15
3.4. Structure of the YOLOv8 architecture	17
3.5. GELAN block in YOLOv9 versus other related methods	18
3.6. Programmable Gradient Information Flow module employed in YOLOv9	18
3.7. Consistent Dual Assignment method integrated into YOLOv10	19
3.8. Partial Self-Attention block implemented in YOLOv10	20
3.9. R-ELAN used in YOLOv12.....	21
3.10. Overview of RT-DETR Architecture	22
3.11. Overview of RF-DETR Architecture	23
3.12. Outdoor setup for data collection using the Fire Test Furnace.	25
3.13. Sample collected data: paired RGB and thermal images.	25
3.14. Custom circular grid calibration board with heating elements	26
3.15. Creation of RGB-T Overlay Images	28
3.16. RGT, RGB, and Thermal datasets' samples across different conditions. .	29
3.17. Workswell WIRIS Enterprise.....	29
3.18. DepthAnything Framework.....	31
3.19. DepthCrafter Framework.....	32
3.20. DepthPro Architecture.....	32
3.21. Marigold Framework.....	33
3.22. Metric3D Framework.....	33
3.23. DPT Model Architecture.....	34

3.24. Classical fire masking pipeline	37
4.1. Overlapping and duplicate fire detections for lightweight YOLO models .	40
4.2. Training convergence (mAP@0.5:0.95) for nano and small YOLO variants.	43
4.3. Precision, Recall, and F1 score as a function of confidence threshold for small YOLO variants.	44
4.4. Training convergence (mAP@0.5:0.95) of YOLOv12 mvst transformer-based models.	4
4.5. Extrinsic calibration visualization showing relative camera poses.	5
4.6. Training convergence (mAP@0.5:0.95) of YOLOv12 across different input modalities.	47
4.7. Example of missed fire by RGB-only model: incipient flames not visually prominent.	4
4.8. Model understanding for fire scenes	8
4.9. Calibration-based metric depth estimation	5
4.10. ZoeDepth-based metric depth estimation	0
4.11. Thermal-based fire source localization	5
4.12. Bottom-center heuristic and classical fire masking for side-propagating fire.	1
4.13. Comparison of classical fire masking and segmentation-based model. . .	5
4.14. Safety rules and constraints logic	2
	5
	2
	5
	3
	5
	3
	5
	3

List of Tables

3.1. IFFS's key system requirements.	12
3.2.DetailsofYOLOv5variants.....	16
3.3.DetailsofYOLOv8variants.....	16
3.4.DetailsofYOLOv9variants.....	17
3.5. DetailsofYOLOv10variants.	19
3.6. DetailsofYOLOv11variants.	20
3.7. DetailsofYOLOv12variants.	21
3.8.DetailsofRT-DETRv2variants.....	22
3.9. DetailsofRF-DETRvariants.....	23
3.10.Summary of state-of-the-art monocular depth estimation models.	34
3.11.Summary of common benchmark datasets for depth estimation.	35
4.1. ResultsforNanoYOLOModelsontheFire,Smoke,andHumandataset.	41
4.2. Results for Small YOLO Models on the Fire, Smoke, and Human dataset.	42
4.3. Performancecomparisonofmediumandtransformer-baseddetectionmodels.	43
4.4. Radial Distortion Coefficients and Uncertainties	46
4.5. PerformanceofYOLOv12strainedondifferentdatasetsforfiredetection.	47
4.6. Comparison of popular monocular depth estimation models.	49

1. Introduction

As the demand for intelligent emergency response systems grows, autonomous drones have emerged as a promising solution for enhancing fire detection and suppression. This chapter outlines the motivation, context, and research objectives behind developing a vision-based Fire Detection and Ranging (FDAR) system tailored for real-time aerial firefighting applications.

1.1. Background and Context

In recent years, the frequency and severity of wildfires and industrial fires have increased significantly due to climate change, urban expansion, and the complexity of industrial operations. These fires can cause devastating damage to ecosystems, infrastructure, and human lives. Rapid and accurate fire detection is essential to initiate timely suppression efforts and minimize the scale of destruction. However, in many cases, fire outbreaks occur in hazardous, remote, or otherwise inaccessible areas where human intervention is either delayed or poses significant risks to emergency personnel.

Autonomous systems, particularly aerial drones, have emerged as promising tools for enhancing firefighting capabilities. Equipped with advanced sensing and processing technologies, drones can provide real-time situational awareness, navigate challenging terrains, and perform surveillance or suppression tasks without endangering human responders. Yet, the effectiveness of such systems depends on the integration of reliable fire, smoke, and human detection capabilities, as well as precise localization methods that allow accurate targeting and safe operation.

This thesis is part of an ongoing research initiative in collaboration with Saxion-SMART lab and IFFS, aimed at developing a fully autonomous firefighting drone. The focus is on building a robust Fire Detection and Ranging (FDAR) system that combines vision-based detection with spatial awareness to identify and localize fire, smoke, and human presence in real time. By leveraging RGB and thermal imaging, and processing the data within the ROS2 framework, the goal is to enable intelligent decision-making for autonomous fire suppression in dynamic and high-risk environments. .

1.2. Problem Statement

Fires in industrial and natural environments often escalate rapidly, leaving little time for traditional firefighting methods to respond effectively. The deployment of human firefighters in such scenarios can be delayed due to safety concerns, limited accessibility, or insufficient situational awareness. While aerial drones have the potential to enhance firefighting operations, their practical application remains limited by the lack of reliable systems for detecting, localizing, and responding to fire incidents autonomously.

The core problem addressed in this thesis is the development of an integrated system capable of real-time fire, smoke, and human detection and localization for autonomous firefighting drones. This involves several interrelated technical challenges. First, visual detection of fire and smoke is highly sensitive to environmental conditions such as lighting, occlusion, and the presence of smoke or reflective surfaces. Human detection is also critical to ensure safety, particularly to prevent accidental harm from suppression mechanisms such as water cannons.

Another major challenge is the estimation of depth from camera imagery, which is essential for determining the precise 3D location of detected elements. This becomes even more complex when relying on monocular cameras or attempting to combine data from heterogeneous sources like RGB and thermal sensors. Additionally, the system must operate within the computational constraints of an onboard drone platform while maintaining real-time performance.

Addressing these challenges is essential for enabling fully autonomous firefighting operations. A system that can accurately detect and localize fire and human presence in three dimensions—and make intelligent decisions accordingly—has the potential to significantly improve response times, reduce risk to human life, and increase the overall effectiveness of emergency interventions in fire-prone environments.

1.3. Research Questions

This thesis addresses the following research questions:

- How can fire, smoke, and human presence be reliably detected in real time using RGB and thermal imagery?
- What are the trade-offs between different object detection architectures in terms of accuracy, speed, and resource efficiency?
- How does the fusion of thermal and RGB data affect detection performance, particularly in low-visibility or occluded conditions?

- Which depth estimation methods are most suitable for accurately localizing detected elements in 3D space using monocular or RGB–thermal inputs?
- How can the system ensure safe fire suppression by evaluating the proximity of humans to fire sources?

1.4. Contribution of the Thesis

This thesis contributes to the development of autonomous firefighting drone technology by designing and evaluating a vision-based Fire Detection and Ranging (FDAR) system.

The main contributions are as follows:

- Comparative analysis of different object detection strategies, including CNN and DETRs, with a focus on real-time performance and accuracy.
- Integration of RGB-T image fusion to enhance detection robustness in challenging conditions such as heavy smoke, low visibility, or occlusion.
- Evaluation of various monocular depth estimation techniques and exploration of stereo calibration between RGB and thermal cameras for spatial localization.
- Implementation of a lightweight fire targeting module that isolates the fire source within the detection region to guide the water cannon accurately.
- Design of a decision logic framework to ensure human safety by preventing fire suppression when a person is detected within or near the fire zone.
 - Deployment of the system within a ROS2 framework, enabling modularity, fast processing, and compatibility with autonomous navigation and planning components.

1.5. Structure Overview

The remainder of this thesis is organized as follows:

- Chapter 2: Literature Review – Provides an overview of related work in fire detection, thermal and RGB image fusion, object detection architectures, depth estimation methods, and autonomous firefighting systems.
- Chapter 3: Methodology – Describes the system architecture, including the detection, depth estimation, and fire targeting modules. It details the dataset preparation, model selection criteria, and implementation within the ROS2 framework.
- Chapter 4: Results and Discussion – Presents experimental results and comparative evaluations of detection models, RGB-T fusion performance, depth estimation accuracy, and the effectiveness of the fire targeting module.

– Chapter 5: Conclusion and Future Work – Summarizes the key findings, discusses the limitations of the current approach, and outlines potential directions for future improvements and integration with autonomous drone navigation and planning systems.

This structure is designed to guide the reader through the background, development, and evaluation of the proposed fire detection and ranging system for firefighting drones.

2. Literature Review

This chapter reviews the current state of research in fire, smoke, and human detection, as well as depth estimation techniques relevant to autonomous firefighting systems. Emphasis is placed on practical deployment constraints, sensor fusion strategies, and model performance in real-world environments.

2.1. Fire and Smoke Detection Methods

Fire and smoke detection is a critical component of autonomous firefighting systems, with its effectiveness heavily influenced by the range of observation and the level of activity within the environment. The detection strategies employed must be adapted to address the specific visual and contextual challenges presented by each operational scenario. This section categorizes existing research into four primary scenarios: short-range and long-range environments, each under low and high activity conditions 2.1.



Figure 2.1: Examples of fire detection tasks at a short (top) and long (bottom) range.

2.1.1. Short-Range Fire Detection with Low Activity

Short-range, low-activity environments—such as warehouses, tunnels, and isolated industrial facilities—offer relatively favorable conditions for fire detection. In these scenarios, cameras have a constrained field of view, and both flames and smoke are generally visi-

ble with minimal occlusion. Traditional fire detection methods relying on handcrafted visual features such as color, shape, and motion have proven effective in this context. Early techniques employed color-based thresholds in various color spaces (e.g., RGB, HSI, YCbCr) [3, 13], while more recent approaches have adopted deep convolutional neural networks (CNNs) for image-level classification or combined detection and segmentation [25, 28, 31]. Lightweight algorithms such as background subtraction remain relevant for edge deployment, offering efficient performance with reduced computational demands.

2.1.2. Long-Range Fire Detection with Low Activity

In long-range, low-activity environments, typical of large industrial site monitoring, direct observation of flames is often limited, especially during early fire stages. Smoke detection becomes the primary focus, although this introduces new challenges due to the visual similarity of smoke with clouds, fog, and other atmospheric conditions. High-resolution imaging is essential to maintain detection accuracy over large distances. Both traditional and deep learning-based methods have been proposed, including background subtraction, semantic segmentation, and object detection using CNN or transformer-based architectures [8, 14, 29]. Attention mechanisms and hybrid models have shown promise in enhancing detection reliability under these visually ambiguous conditions.

2.1.3. Short-Range Fire Detection with High Activity

Short-range, high-activity environments—such as factory floors, public venues, and busy storage facilities—present additional complexity due to frequent movement and visual clutter. Dynamic elements like vehicles, machinery, or people may cause occlusions or be mistakenly identified as fire due to color and motion artifacts. To mitigate these issues, temporal analysis has been widely adopted. Frame-to-frame comparisons, flame dynamics modeling, and multi-frame feature extraction are used to distinguish actual fire from transient disturbances [19, 27]. More recent methods incorporate temporal deep learning models, such as LSTMs and 3D CNNs, which capture both spatial and temporal features [9, 2]. These approaches enhance detection robustness in fast-changing environments.

2.1.4. Long-Range Fire Detection with High Activity

The most challenging scenario involves detecting fire at a distance in highly dynamic environments, such as construction sites or industrial transport hubs. Visual occlusions, moving objects, artificial lighting, and airborne particles contribute to high false positive rates. Addressing these challenges requires sophisticated models that integrate spatial and temporal analysis, multi-modal sensing (e.g., RGB and thermal), and advanced



Figure 2.2: Examples of challenges in flame and smoke recognition.

architectures like CNN-LSTM hybrids or transformer-based detectors [1, 7]. Region proposal mechanisms, contextual scene understanding, and multi-sensor fusion have also been explored to improve localization and reliability. Despite these advances, this scenario continues to demand further research to improve accuracy and minimize false detections in real-world applications.

2.2. Human Detection

Human detection is a key requirement in autonomous firefighting and rescue operations, particularly in environments with limited visibility due to smoke or fire. Manual search methods are often dangerous and inefficient, making camera-based detection systems a safer alternative. Thermal and optical cameras are commonly used, each with unique strengths and limitations. As shown in Fig. 2.3, optical cameras struggle in dense smoke (a), while thermal cameras remain effective by detecting heat signatures (c). However, in the presence of nearby heat sources (b), thermal detection accuracy may degrade. Fusing optical and thermal data (d) can improve overall detection performance. Accurate human localization is essential not only for rescue but also to prevent autonomous drones from targeting suppression systems toward people.

2.2.1. Single Sensor Approaches

Optical Cameras Optical cameras are commonly utilized for UAV-based human detection. Techniques such as real-time detection [26] and depth estimation [10] improve accuracy in standard environments. Feature extraction methods, such as rotation-invariant histogram of oriented gradients (RIHOG), combined with machine learning models like support vector machines (SVMs), have also demonstrated high precision rates [20]. However, optical

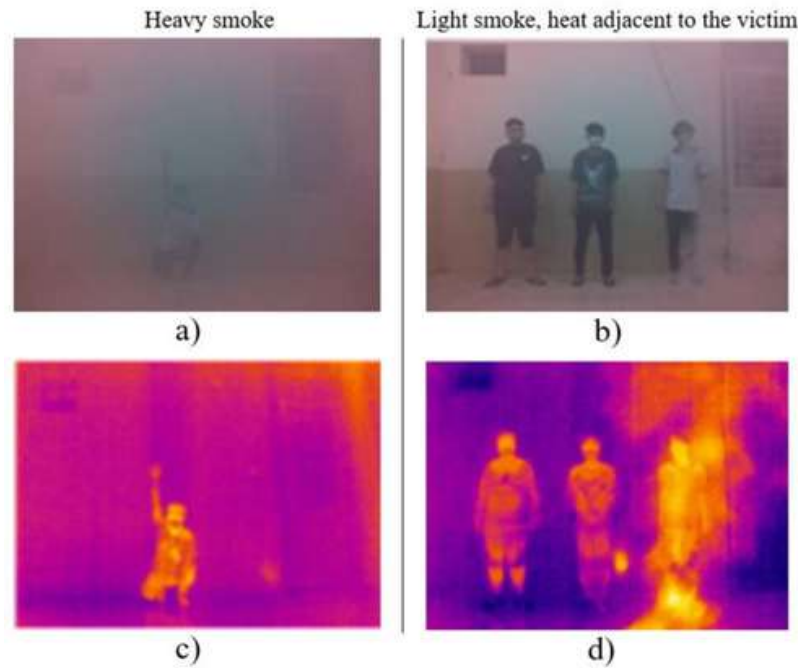


Figure 2.3: Comparison between optical and thermal cameras

cameras perform poorly in high-smoke conditions, where visibility below 1m renders detection ineffective.

Thermal Cameras Thermal imaging is widely employed for human detection by capturing heat signatures emitted from the human body. Work like [30] and [12] demonstrate segmentation methods for detecting human limbs in thermal images, utilizing mathematical morphological operations. Also [18] apply a simple thresholding approach to identify high-temperature regions indoors and outdoors, though accuracy declines in fire smoke-contaminated environments. Similarly, [24] propose two detection algorithms based on black body radiation and background subtraction, achieving accuracies of 69% and 79%, respectively, depending on environmental factors. Other methods focus on thermal pixel value analysis or classifier-based approaches such as Haar features, achieving varying levels of accuracy depending on distance and object orientation.

Deep learning techniques, such as convolutional neural networks (CNNs), have significantly enhanced thermal-based human detection. work like [15] improve detection performance using a YOLOv3-based CNN model, while in [21] achieve an average precision of 82.49% with MobileNet. High precision rates are also reported using YOLOv4, SSD, and other CNN-based models [16]. In extreme conditions such as smoke-filled environments, thermal imaging remains robust, with studies showing improved detection through image processing techniques and fusion-based Bayesian estimation.

2.2.2. Multi-Sensor Approaches

Combining thermal and optical imaging enhances human detection capabilities. In [11] present a framework for fusing thermal and RGB images to improve human shape extraction. Also, [17] introduce an algorithm for real-time multi-human detection by analyzing histogram correlations between thermal and optical data. Further, [5] integrate infrared cameras for human presence detection while using optical sensors for scene visualization. Shape template matching techniques have also been utilized in real-time UAV applications, where RGB and thermal images are blended to improve accuracy [6].

Overall, while optical cameras provide high-resolution imagery, thermal cameras demonstrate superior performance in adverse conditions, particularly in fire and smoke environments. The integration of multiple sensor modalities further enhances detection reliability, making it a promising approach for autonomous firefighting drones.

2.3. Depth Estimation Techniques

Depth estimation is a critical capability for autonomous systems operating in unstructured or hazardous environments, such as those affected by fire. Accurate 3D localization of detected elements—fire, smoke, or human—is essential for effective decision-making and safe actuation, particularly when targeting suppression mechanisms. Two main approaches are commonly used for depth estimation: monocular depth estimation and stereo vision.

2.3.1. Monocular Depth Estimation

Monocular depth estimation aims to infer scene depth from a single RGB image using learned visual cues Fig. 2.4. Recent advancements in deep learning have significantly improved the performance and generalizability of monocular depth models. Transformer-based and convolutional architectures such as DPT (Dense Prediction Transformer) [23], Depth Anything [4], and MiDaS [22] have demonstrated strong performance across diverse datasets. These models are trained on large-scale multi-scene datasets and utilize encoder-decoder structures with multi-scale supervision, enabling them to generalize well in unseen environments.

Compared to stereo methods, monocular models are computationally lightweight and easier to deploy on embedded systems, making them suitable for drone applications. However, they can struggle with absolute depth accuracy and may require post-processing or scale correction when precise localization is required.



Figure 2.4: Monocular Depth Estimation Examples

2.3.2. Stereo Depth Estimation and RGB–Thermal Calibration

Stereo vision uses two cameras—typically with a fixed baseline—to triangulate depth based on disparity between the images. While traditional stereo setups use identical RGB cameras, there is growing interest in using heterogeneous sensor pairs such as RGB and thermal cameras for stereo matching in low-visibility environments. This approach requires careful calibration to align the differing spectral responses and resolutions of the two sensors.

Recent work explores cross-modal stereo matching and registration techniques to enable depth estimation between RGB and thermal image pairs. While more computationally intensive, stereo methods can provide metrically accurate depth, making them valuable when precision is critical, such as in fire source targeting.

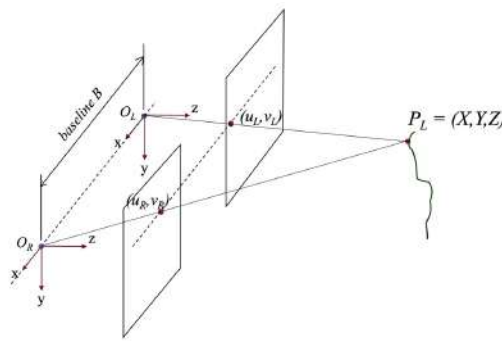


Figure 2.5: Stereo and Disparity Geometry

Overall, the choice between monocular and stereo techniques depends on the system’s operational constraints. For real-time drone deployment in fire scenarios, monocular depth estimation offers lightweight and adaptable solutions, while stereo methods—particularly RGB–thermal fusion—can enhance spatial accuracy where conditions permit.

3. Methodology

This chapter outlines the design and implementation of the Fire Detection and Ranging (FDAR) system developed for autonomous firefighting drones. It details the system architecture, individual module functionalities, integration strategies, and experimental setup used to evaluate performance under realistic conditions.

3.1. Problem Formulation

The primary objective of this thesis is to develop an integrated Fire Detection and Ranging (FDAR) system for autonomous firefighting drones, according to IFFS's requirements

3.1. This system must be capable of detecting and localizing fire, smoke, and human presence in real-time, even in challenging environments characterized by low visibility, occlusion, or high environmental activity. The system must not only identify hazardous elements but also produce accurate three-dimensional localization to enable effective and safe suppression operations.

The FDAR system takes as input synchronized RGB and thermal imagery captured from onboard sensors. The desired output consists of:

- A set of detections corresponding to fire, smoke, and human presence, each with associated confidence scores.
- Depth information for each detection, providing 3D coordinates relative to the drone's frame of reference.
- Safety flags indicating whether fire suppression actions, such as activating the water cannon, can be safely executed, ensuring no human is present within the suppression zone.

The core challenges include:

- Robust detection of fire and smoke features under varying visibility conditions, including dense smoke and occlusion.
- Accurate and low-latency human detection for safety assurance, using RGB, thermal, or fused imagery.

- Precise depth estimation in real-time, based on monocular methods or stereo calibration, to support accurate targeting.
- Efficient integration of all components within the constraints of onboard computation and power resources.

In summary, this chapter presents a comprehensive description of the system architecture, detailing the detection, depth estimation, and fire targeting modules. It further explains the safety logic for suppression decisions, outlines implementation details, and specifies the evaluation metrics and experimental setup used to validate system performance.

Table 3.1: IFFS’s key system requirements.

Requirement Category	Description
DetectionAccuracy	The system must detect potential fire sources with high accuracy and distinguish fire from other heat sources to prevent false positives.
FireClassification	The system should classify fire states: no fire, small fire (extinguishable), and large fire (non-extinguishable).
RangingCapabilities	The system must determine the distance to the fire and estimate its size to categorize fire intensity.
SafetyMeasures	A human detection module must prevent accidental activation of fire suppression mechanisms when a person is present.
Communication	The system should support wired communication with the UAV and wireless communication with the Ground Control Station (GCS).
Processing	AI-based processing should enable real-time fire detection and classification with minimal computational latency.
EnvironmentalAdaptability	The system must operate in diverse conditions, including smoke, fog, and varying lighting, and withstand extreme temperatures.
HardwareConstraints	The system must integrate multiple sensors (thermal, RGB, LiDAR, ToF), not exceed 25W power consumption, and remain under 1kg to ensure UAV stability.

3.2. System Architecture Overview

The proposed Fire Detection and Ranging (FDAR) system is designed to operate as an integrated perception module onboard an autonomous firefighting drone. Its architecture is modular and scalable, leveraging the Robot Operating System 2 (ROS2) framework for real-time communication and interoperability among components. The system architecture comprises three primary modules: the Detection Module, the Depth Estimation Module, and the Fire Targeting Module, all of which interact to provide accurate and timely localization of fire, smoke, and human presence.

3.2.1. Module Interactions and Data Flow

The system begins with data acquisition from synchronized RGB and thermal cameras mounted on the drone platform. The raw sensor data is processed through the Detection Module, which performs object detection to identify and classify fire, smoke, and human presence. Depending on environmental conditions, the module can operate with RGB images, thermal images, or a fusion of both modalities to enhance detection robustness. The outputs from the Detection Module, which include bounding boxes and class labels, are forwarded to the Depth Estimation Module. This module estimates the distance to each detected object; using monocular depth estimation methods. The resulting 3D

coordinates are then associated with the respective detections to provide spatial awareness. The Fire Targeting Module refines the fire detection results by applying classical computer vision techniques to isolate the fire source within the detected bounding box. It evaluates the presence of humans in proximity to the fire and determines whether suppression mechanisms can be safely activated. A safety flag is generated to indicate whether the fire is safe for intervention. Figure 3.1 illustrates the complete data flow across the FDAR system components, showing the interaction between modules from input acquisition to safety-aware output generation.

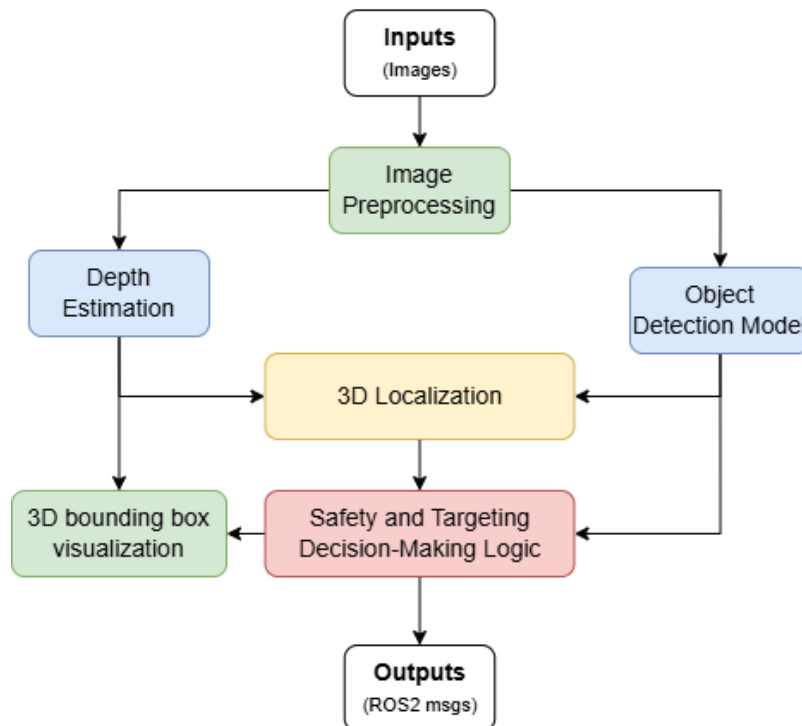


Figure 3.1: FDAR module workflow diagram

3.2.2. ROS2 Framework Integration

Each module is encapsulated as a ROS2 node, enabling flexible deployment, modular design, and real-time communication between components. The use of ROS2 ensures compatibility with existing drone control frameworks and allows seamless integration of perception, navigation, and cannon triggering subsystems. The system publishes the final outputs, including:

- Object class (e.g., fire, human, smoke),
- 3D position estimated using monocular depth,
- Detection confidence,

- Bounding box coordinates,
- Safety flags (e.g., proximity to humans).

3.2.3. System Workflow Summary

The system architecture ensures that detection, depth estimation, and decision-making operate concurrently and efficiently. Data flows from the camera through the detection and depth modules, enabling accurate 3D localization and safe actuation. This modular design facilitates scalability to different platforms and environmental conditions.



Figure 3.2: FDAR System GUI

3.3. Detection Module

The detection module serves as the system’s primary component for identifying and classifying fire, smoke, and human presence in captured imagery. This section details the dataset used for training and evaluating detection models, model selection criteria, and comparative evaluations.

3.3.1. Dataset Description

The dataset employed in this study is the Fire, Human, and Smoke dataset, comprising 9,749 annotated RGB images curated for the specific purpose of multi-class detection. Collected from a variety of sources, including surveillance cameras and both aerial and ground-level viewpoints, the dataset captures a broad spectrum of real-world environments. It includes images recorded under diverse conditions—daytime and nighttime lighting, varying distances, scenes with and without occlusion, and a range of fire and smoke

intensities and shapes. This variability ensures that detection models trained and evaluated on this dataset are exposed to realistic and challenging scenarios.

The dataset includes a total of 19,907 annotations, with 9,439 instances of fire, 6,032 instances of humans, and 4,436 instances of smoke, providing a substantial volume of labeled data for robust model development. These annotations facilitate the evaluation of models across different detection classes, contributing to a comprehensive performance assessment. Fig. 3.3 illustrates examples of the dataset's diversity. The dataset is openly accessible through Roboflow, offering a valuable resource for research and development in fire and smoke detection.

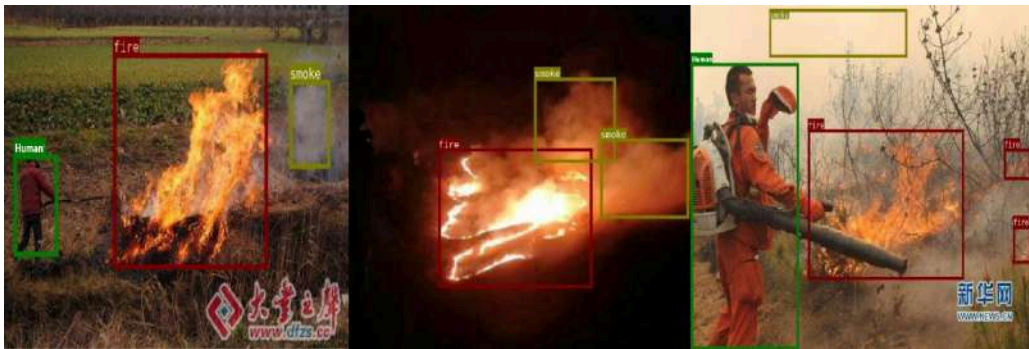


Figure 3.3: Sample images from the Fire, Human, and Smoke dataset.

3.3.2. Model Selection and Architecture

This subsection presents a comprehensive review of the detection models explored for this research, focusing on their architectures, innovations, and suitability for fire, smoke, and human detection. The models analyzed include YOLOv5, YOLOv8, YOLOv9, YOLOv10, YOLOv11, and YOLOv12, with each offering unique advancements in performance, efficiency, and architectural design.

YOLOv5

YOLOv5 represents a robust baseline architecture in the YOLO family, offering a modular design comprising a backbone, neck, and head. The backbone utilizes CSPDarknet53 for feature extraction, while the neck incorporates Path Aggregation Network (PANet) and Spatial Pyramid Pooling Fast (SPPF) modules to refine multi-scale feature aggregation. The detection head employs anchor-based bounding box prediction and class confidence scoring. YOLOv5's widespread use and balance of detection performance and computational efficiency make it a practical choice for real-time applications on embedded platforms, despite newer YOLO variants offering more advanced features. YOLOv5 is available in

five model variants, each offering different trade-offs between model size, parameter count, and computational cost. The variant details are summarized in Table 3.2.

Table 3.2: Details of YOLOv5 variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
YOLOv5n	64	1.9	4.5
YOLOv5s	0	7.2	16.5
YOLOv5m	64	21.2	49.0
YOLOv5l	0	46.5	109.1
YOLOv5x	64	87.7	205.7

YOLOv8

YOLOv8, shown in Fig. 3.4, a successor to YOLOv5, introduces a range of architectural and functional upgrades. Notably, it transitions to an anchor-free detection strategy, focusing on object centers rather than anchor boxes. The model replaces the C3 bottleneck module with C2f blocks, which expand the receptive field and capture more contextual information. Mosaic augmentation is incorporated during training to improve generalization. While it retains the backbone and neck structures similar to YOLOv5—including CSPDarknet53, PANet, and SPPF—YOLOv8 optimizes information flow and introduces decoupled heads for enhanced detection. These modifications lead to improved accuracy and inference speed across diverse detection scenarios. YOLOv8 is available in five variants, whose characteristics are summarized in Table 3.3.

Table 3.3: Details of YOLOv8 variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
YOLOv8n	64	3.2	8.7
YOLOv8s	0	11.2	28.6
YOLOv8m	64	25.9	78.9
YOLOv8l	0	43.7	165.2
YOLOv8x	64	68.2	257.8

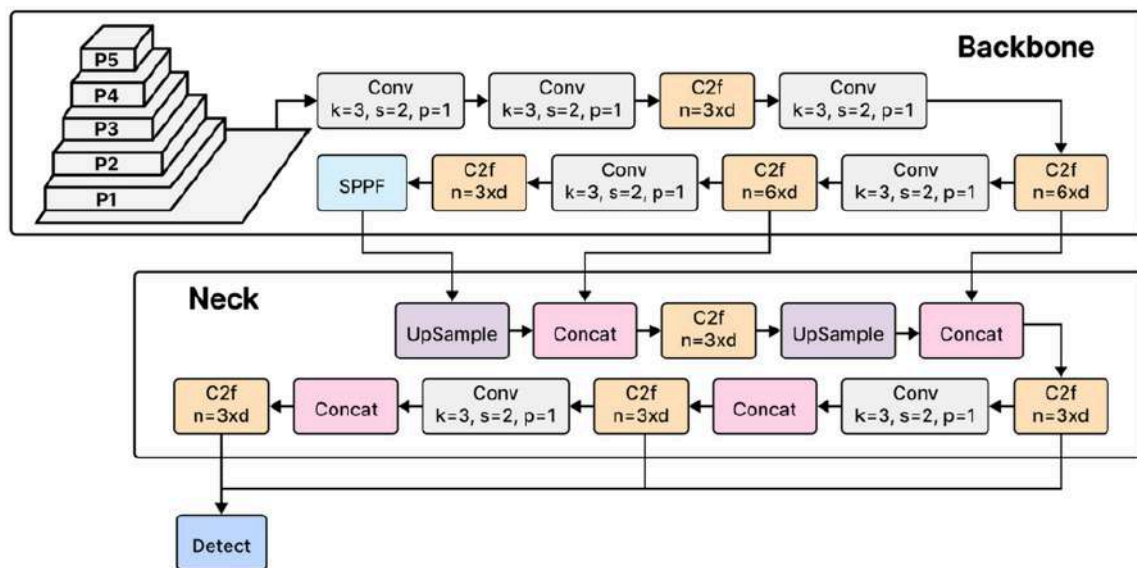


Figure 3.4: Structure of the YOLOv8 architecture

YOLOv9

YOLOv9 builds upon previous YOLO architectures by addressing information loss within compressed network regions. It introduces reversible functions to retain input information, Programmable Gradient Information (PGI) modules for efficient gradient flow during training, and the Generalized Efficient Layer Aggregation Network (GELAN) for effective feature aggregation. GELAN combines principles from CSPNet and ELAN architectures to enable both lightweight design and accurate detection, Fig. 3.5. PGI introduces a dual-branch architecture, shown in Fig. 3.6, with the main branch used for inference and the secondary branch enhancing training efficiency. These innovations collectively improve YOLOv9’s detection performance while maintaining computational efficiency. YOLOv9 is available in multiple variants designed for different performance and resource requirements. Table 3.4 summarizes the key characteristics.

Table 3.4: Details of YOLOv9 variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
YOLOv9t	64	2.0	7.7
YOLOv9s	0	7.1	26.4
YOLOv9m	64	20.0	76.3
YOLOv9l	0	25.3	102.1
YOLOv9x	64	57.3	189.0

0

64

0

64

0

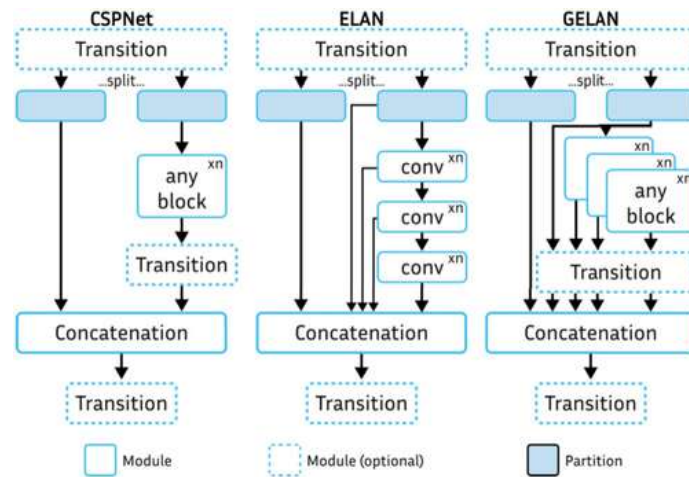


Figure 3.5: GELAN block in YOLOv9 versus other related methods

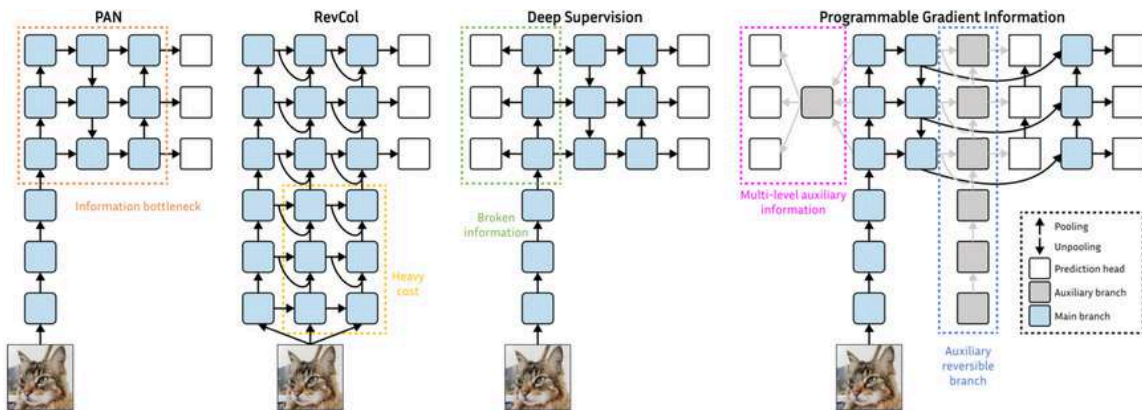


Figure 3.6: Programmable Gradient Information Flow module employed in YOLOv9

YOLOv10

YOLOv10 focuses on overcoming the limitations of Non-Maximum Suppression (NMS) and enhancing inference speed. It introduces a one-to-one assignment strategy, where each ground-truth instance is matched with a single prediction, eliminating redundant detections and reducing post-processing latency. This strategy is complemented by consistent dual assignments during training, allowing the model to leverage both one-to-one and one-to-many assignments while retaining efficiency during inference, Fig. 3.7. YOLOv10 also employs compact inverted blocks (CIB) for computational efficiency and partial self-attention (PSA) modules to enhance contextual feature extraction, Fig. 3.8. By decoupling spatial downsampling and channel transformations, YOLOv10 achieves reduced computational complexity and faster processing. YOLOv10 includes six model variants tailored to different deployment scenarios. Their details are presented in Table 3.5.

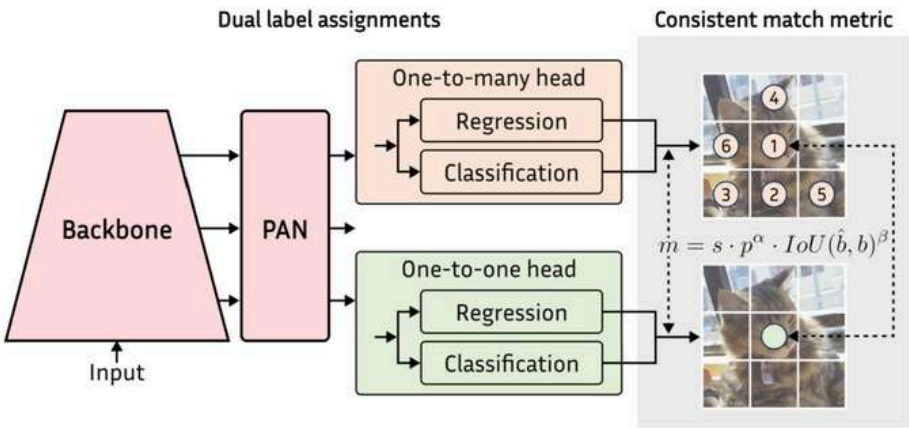


Figure 3.7: Consistent Dual Assignment method integrated into YOLOv10

Table 3.5: Details of YOLOv10 variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
YOLOv10n	64	2.3	6.7
YOLOv10s	0	7.2	21.6
YOLOv10m	64	15.4	59.1
YOLOv10b	0	19.1	92.0
YOLOv10l	64	24.4	120.3
YOLOv10x	0	29.5	160.4

YOLOv11

YOLOv11, developed by the same team behind YOLOv5 and YOLOv8, introduces notable architectural enhancements. The model adopts C3k2 blocks within the backbone and head, replacing the C2f modules of YOLOv8. C3k2 blocks split feature maps and process them through dual convolutional paths, optimizing efficiency and maintaining accuracy. YOLOv11 also introduces the Cross Stage Partial with Spatial Attention (C2PSA) module, which emphasizes relevant spatial regions, improving detection of small and occluded objects. The architecture retains key features from YOLOv8, including the SPPF module and decoupled heads, while refining feature extraction and aggregation through innovative block structures. YOLOv11 provides five variants, as detailed in Table 3.6.

YOLOv12

YOLOv12 marks a significant advancement in the YOLO family, introducing an attention-centric architecture that diverges from traditional CNN-based designs while preserving

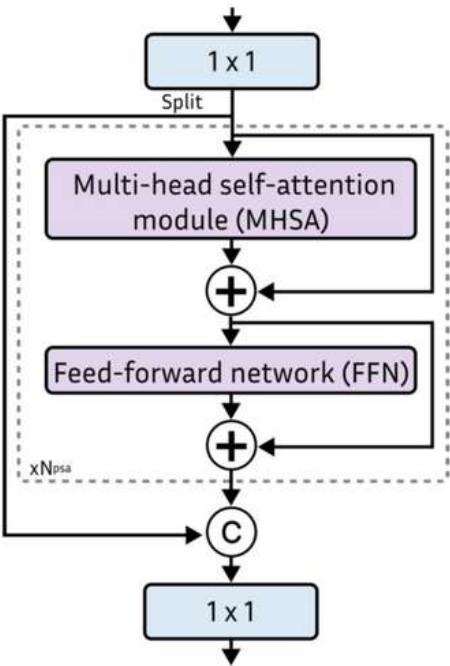


Figure 3.8: Partial Self-Attention block implemented in YOLOv10

Table 3.6: Details of YOLOv11 variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
YOLOv11n	64	2.6	6.5
YOLOv11s	0	9.4	21.
YOLOv11m	64	20.1	5
YOLOv11l	0	25.3	68.
YOLOv11x	64	56.9	194.9
	0		86.

the real-time inference capability essential for practical deployment. Central to its architecture is the Area Attention Mechanism, which efficiently captures large receptive fields by dividing feature maps into equally sized regions, reducing computational costs and maintaining effective receptive field size. The model also incorporates Residual Efficient Layer Aggregation Networks (R-ELAN) for optimized feature aggregation and enhanced information flow, along with an Optimized Attention Architecture that includes FlashAttention for memory efficiency, no positional encoding for streamlined processing, and adaptive MLP ratios for improved balance between attention and feed-forward layers. Additionally, YOLOv12 integrates 7×7 separable convolutions as a position perceiver to encode spatial information. The latest YOLOv12 architecture also includes multiple variants, as outlined in Table 3.7.

Table 3.7: Details of YOLOv12 variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
YOLOv12n	64	2.6	6.5
YOLOv12s	0	9.3	21.4
YOLOv12m	64	20.2	67.5
YOLOv12l	0	26.4	88.9
YOLOv12x	64	59.1	199.0

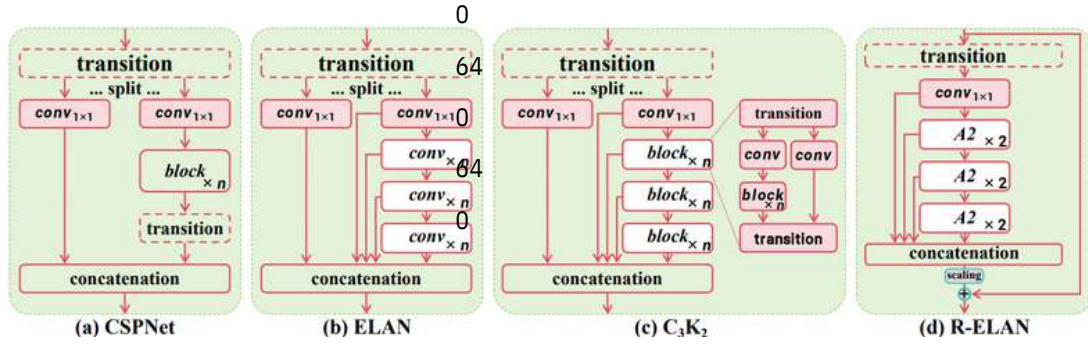


Figure 3.9: R-ELAN used in YOLOv12

RT-DETRv2

RT-DETRv2 (Real-Time Detection Transformer v2) builds upon the RT-DETR architecture to further improve accuracy while preserving real-time performance. Like its predecessor, RT-DETRv2 follows an end-to-end transformer-based design with a hybrid backbone: a CNN extracts multi-scale features which are then processed by transformer encoder-decoder layers without requiring NMS in post-processing.

Key upgrades in RT-DETRv2 center on the transformer decoder’s deformable attention mechanism. The model allocates a different number of sampling points to each feature scale in the decoder’s attention, a technique called selective multi-scale sampling, to better capture objects of varying sizes. This scale-aware attention, combined with the original uncertainty-based query selection mechanism, enhances detection of both small and large targets.

RT-DETRv2 also introduces an optional discrete sampling operator (replacing the continuous grid sampling) to ease deployment on diverse hardware. Additionally, it adopts improved training strategies—such as dynamic data augmentation and scale-adaptive hyperparameter tuning—to boost performance (achieving higher mAP than RT-DETR) without sacrificing inference speed. RT-DETRv2 is provided in multiple model sizes to cater to different resource settings, as summarized in Table 3.8.

Table 3.8: Details of RT-DETRv2 variants.

Model	Size (pixels)	Params (M)	FLOPs (B)
RT-DETRv2-s	64	2	60
RT-DETRv2-m	0	0	100
RT-DETRv2-l	64	3	136
RT-DETRv2-x	0	6	259
	64	4	

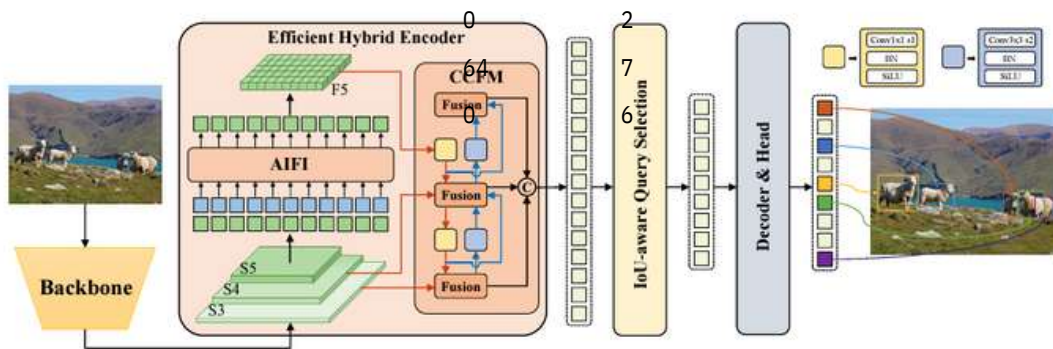


Figure 3.10: Overview of RT-DETR Architecture

RF-DETR

RF-DETR (Roboflow DETR) is a state-of-the-art real-time detector introduced in 2025 that achieves unprecedented accuracy (exceeding 60% mAP on COCO) while maintaining low latency. It belongs to the DETR family of transformer-based models and emphasizes both high precision and adaptability to various domains.

Architecturally, RF-DETR builds on the Deformable DETR framework, combining a transformer decoder with a powerful vision transformer backbone. Notably, it leverages a pre-trained DINOv2 Vision Transformer as its backbone, enabling the model to harness rich, generalizable features learned from large-scale self-supervised training. This design choice greatly enhances RF-DETR's ability to generalize to new tasks (such as detecting fire, smoke, or humans in novel environments) and small datasets without sacrificing speed.

Unlike the original Deformable DETR which uses multi-scale feature maps, RF-DETR opts for a single-scale feature extraction from the backbone, simplifying the pipeline and improving efficiency. The detector remains anchor-free and NMS-free: object queries directly yield final detections in one-to-one fashion, which avoids duplicate boxes and additional post-processing overhead.

Despite its transformer complexity, RF-DETR is optimized for real-time operation on edge hardware. Table 3.9 summarizes the specifications of the RF-DETR variants.

Table 3.9: Details of RF-DETR variants.

Mo del	Size (pixels)	Params (M)	FLOPs (B)
RF-DETR-Base	64	29	90*
RF-DETR-Large	0	128	340*

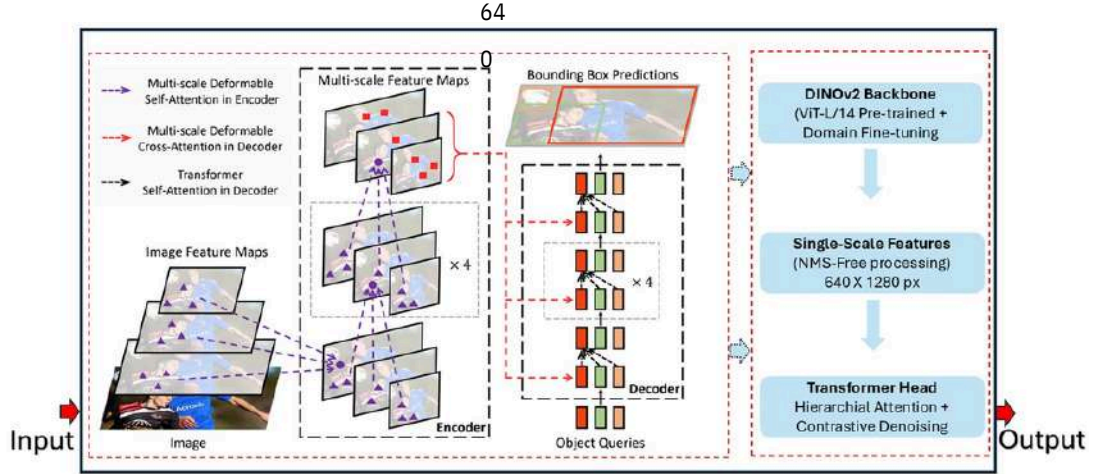


Figure 3.11: Overview of RF-DETR Architecture

3.3.3. Performance Evaluation

The evaluation of the detection models in this study was conducted using standard performance metrics commonly applied in object detection research. These include precision, recall, and mean average precision (mAP), each providing a unique perspective on model effectiveness. Below is a detailed description of these metrics.

Precision

Precision measures the proportion of correctly identified positive instances among all instances predicted as positive by the model. It reflects the model's ability to avoid false positives, ensuring that detections labeled as fire, smoke, or human are accurate. Precision is mathematically expressed as:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3.1)$$

where TP denotes true positives and FP represents false positives. A higher precision value indicates that the model successfully minimizes the misclassification of negative instances as positives.

Recall

Recall assesses the model's capability to detect all relevant positive instances within the dataset. It quantifies the proportion of true positives detected out of the total actual positives. In other words, recall reflects how effectively the model captures all instances of fire, smoke, and human presence. The formula for recall is:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3.2)$$

where FN denotes false negatives. A high recall value signifies that the model accurately identifies the majority of true positive instances, reducing the risk of missed detections.

Mean Average Precision

Mean Average Precision (mAP) provides a comprehensive evaluation metric by balancing precision and recall across various classes. It computes the average precision (AP) for each class by calculating the area under the precision-recall curve and then averages these values across all classes. The AP is defined as:

$$AP = \int_0^1 \text{precision}(r) dr \quad (3.3)$$

where $\text{precision}(r)$ indicates the precision at a specific recall level r . The overall mAP score is obtained by averaging AP scores over all classes. Different variations of mAP are often used depending on the intersection over union (IoU) threshold. For example, $\text{mAP}@0.5$ measures performance at an IoU threshold of 0.5, while $\text{mAP}@0.5:0.95$ evaluates detection across multiple thresholds ranging from 0.5 to 0.95. A higher mAP indicates superior detection performance by effectively balancing precision and recall.

3.3.4. Fusion of Thermal and RGB Imagery

The integration of thermal and RGB images offers significant advantages in fire, smoke, and human detection, especially in challenging environmental conditions. This subsection details the steps taken to calibrate disparate camera types, construct a robust multi-sensor system, and leverage the complementary strengths of both modalities.

Fire Dataset Collection and Outdoor Setup

To develop and evaluate our RGB-T fusion pipeline under realistic firefighting scenarios, we collected a diverse fire dataset using a custom-built outdoor test setup centered around a standardized Fire Test Furnace. This setup included dual thermal and RGB cameras mounted on a calibrated tripod system (Figure 3.12), aligned using a stereo

calibration process. The test rig was complemented by a suppression system including a fire extinguisher and a real-time monitoring station, enabling controlled ignition and capture of fire events from various angles and lighting conditions.

During data collection, we specifically targeted difficult edge cases: partially visible flames, small or starting fires (clearly visible in thermal but hard to detect in RGB), and common distractors such as red-colored non-flammable objects that visually resemble fire. Figure 3.13 presents frames sample from the collected dataset, showing both RGB and thermal views of an active fire in the furnace. This careful scene design ensured our dataset supports robust training and evaluation of fusion-based detection strategies.



Figure 3.12: Outdoor setup for data collection using the Fire Test Furnace.



Figure 3.13: Sample collected data: paired RGB and thermal images.

Custom Checkerboard for Multi-spectral Stereo Calibration

To enable precise calibration between the thermal and RGB cameras, we constructed a custom asymmetric circular grid calibration board equipped with heated elements. The

choice of an asymmetric circular grid was intentional, as it eliminates the 180-degree ambiguity inherent in symmetric patterns like checkerboards. This ensures a unique and unambiguous calibration outcome, even when the board is rotated. The heated elements embedded within the grid provide strong contrast in thermal imagery, making the calibration pattern simultaneously visible in both thermal and RGB domains.



Figure 3.14: Custom circular grid calibration board with heating elements

Calibration Procedure and Code Modification

Accurate calibration of cameras with fundamentally different spectral sensitivities and resolutions presents unique challenges. Neither the standard MATLAB nor OpenCV toolboxes provide out-of-the-box solutions for calibrating a thermal-RGB stereo pair. To address this, we adapted the `estimateCameraParameters.m` and `rectifyStereoImages.m` functions within the MATLAB toolbox to handle images from cameras with different intrinsic properties and resolutions. These modifications allowed us to establish a precise geometric relationship between the thermal and RGB cameras, essential for accurate image alignment and subsequent fusion.

The calibration process involves capturing multiple images of the custom checkerboard from various angles and distances, computing intrinsic and extrinsic parameters for both cameras, and rectifying images to align corresponding points. This alignment enables the creation of RGB-T composite images, where each pixel is enriched with both color and thermal information.

Creation of RGB-T Overlay Images

Using the calibrated parameters, we generated RGB-T images by accurately aligning thermal and RGB imagery based on camera calibration data. The process begins with projecting points from the thermal image into 3D space. Given a point $p_t = (u_t, v_t)$ in the thermal image, its projection into 3D is computed using the inverse of the thermal camera intrinsic matrix K_t^{-1} :

$$P_t = K_t^{-1} \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix} \quad (3.4)$$

where d_t represents the estimated depth for the point, sourced from the depth estimation module described in the next section.

Once the point is represented in the thermal camera's 3D coordinate frame, we apply the extrinsic transformation matrix $[R \quad T]$, derived from the stereo calibration, to map it to the RGB camera coordinate frame:

$$P_r = R P_t + T \quad (3.5)$$

where R is the rotation matrix and T is the translation vector between the thermal and RGB cameras.

Finally, the transformed point P_r is projected onto the RGB image plane using the RGB camera's intrinsic matrix K_r :

$$p_r = K_r \begin{bmatrix} X_r/Z_r \\ Y_r/Z_r \\ 1 \end{bmatrix} \quad (3.6)$$

where (X_r, Y_r, Z_r) are the 3D coordinates of P_r . This projection aligns the thermal information with the RGB image, enabling the creation of RGB-T composite images. The fusion effectively combines color and thermal characteristics on a per-pixel basis, enriching the representation of the scene.

The resulting RGB-T images, augmented with depth data, provide a comprehensive multi-modal view, capturing both spectral and spatial information. This fusion enhances detection capabilities, particularly in complex scenarios where either RGB or thermal data alone would be insufficient.

Advantages of RGB-T Fusion over Single Modalities

RGB-T imagery effectively mitigates the limitations inherent in using RGB or thermal data alone. RGB images, while rich in color detail, are susceptible to false positives caused

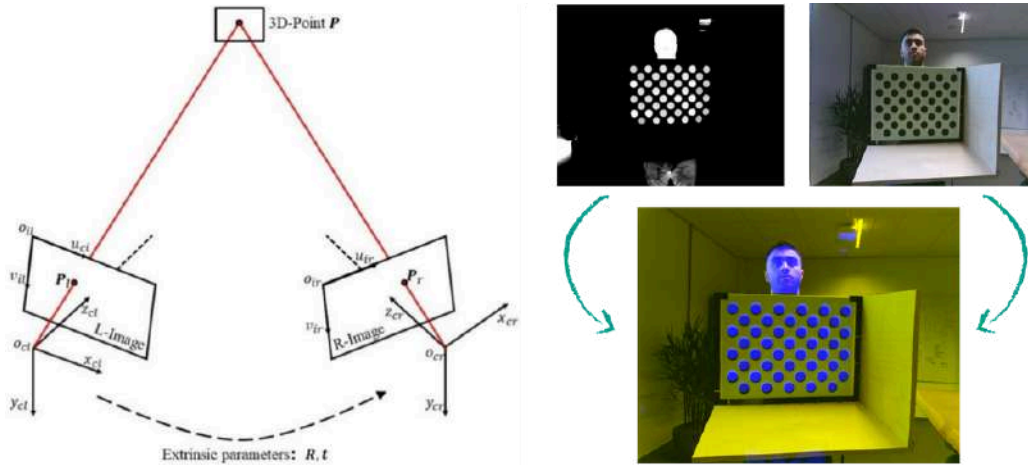


Figure 3.15: Creation of RGB-T Overlay Images

by bright light sources such as street lamps, sunlight, or reflections, and can be obstructed by smoke. Conversely, thermal images are robust against lighting challenges but can be misled by heat sources adjacent to the camera or fail to detect objects obscured by transparent materials that block heat transmission.

To integrate RGB and thermal data, researchers typically employ fusion strategies categorized as early fusion (combining inputs before the network), middle fusion (merging features from separate RGB and thermal branches at intermediate layers), and late fusion (combining final outputs or decision-level features). While middle and late fusion strategies allow deep interaction between modalities, they often come at the cost of increased computational complexity and require multi-stream network designs.

In this work, we adopt a novel early fusion strategy by constructing a three-channel RGT image, where the blue channel in standard RGB input is replaced by thermal data. This approach provides a compact and computationally efficient solution while preserving complementary features from both spectral domains. Specifically, the red and green channels maintain key visual characteristics such as edges and texture, whereas the thermal channel contributes robust heat-based cues, especially valuable under poor illumination, smoke, or cluttered backgrounds. Replacing the blue channel—often the least informative under degraded visibility—helps retain more discriminative signal without increasing input dimensionality, making the fusion naturally compatible with standard CNN architectures.

Figure 3.16 illustrates sample from RGT, RGB, and thermal images across three diverse datasets with varying lighting conditions, perspectives, and environmental complexity, demonstrating the potential of this fusion technique in real-world fire detection scenarios.

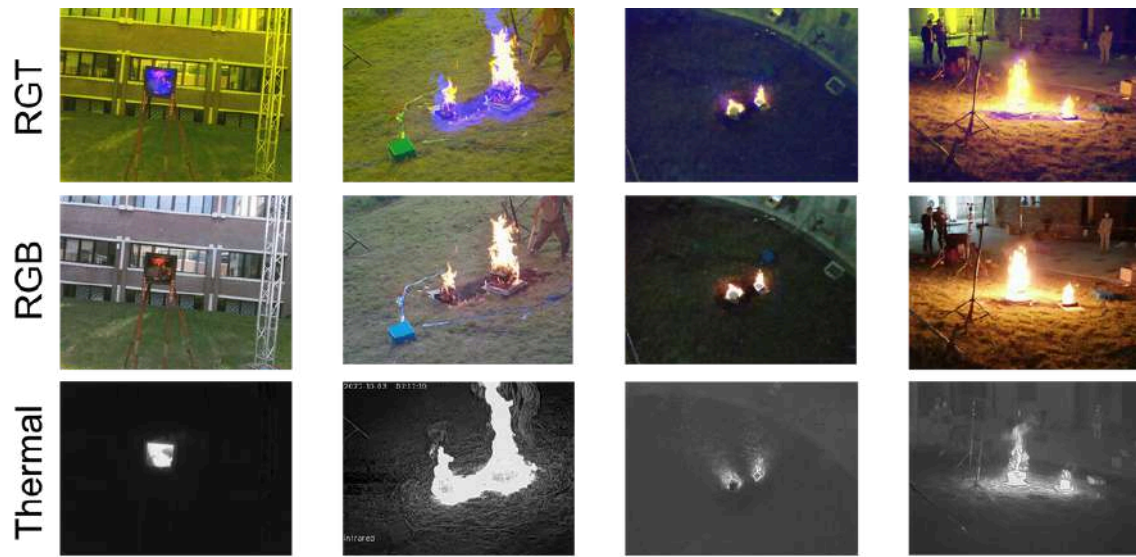


Figure 3.16: RGT, RGB, and Thermal datasets' samples across different conditions.

3.3.5. Camera Specifications: Workswell WIRIS Enterprise

The thermal and RGB images used in this study were captured using the Workswell WIRIS Enterprise camera, an integrated multi-sensor system designed for UAV operations. The thermal sensor operates in the 7.5–13.5 μm spectral range with a native resolution of 640×512 pixels (interpolated up to 1266×1010) and a temperature sensitivity of 30 mK. The RGB module includes a high-resolution 4896 $\times$ 3888 pixel camera and a Full HD optical zoom camera (1920 $\times$ 1080) with up to 30x zoom. Both modules support multiple fields of view and real-time data streaming, making the WIRIS Enterprise a robust choice for synchronized thermal-RGB data acquisition in diverse and challenging environments.



Figure 3.17: Workswell WIRIS Enterprise

3.3.6. Implementation Details

The primary goal of this study is to conduct a comparative evaluation of different detection models in a standardized environment, rather than optimizing individual models. To

maintain consistency, most training parameters were fixed across all experiments. Training was performed using a workstation equipped with four NVIDIA GeForce RTX 4050 GPUs (each with 6 GB memory), supported by 32 GB of system RAM, running on Microsoft Windows 10 with CUDA version 12.8 and NVIDIA driver version 571.96.

The dataset was divided into a training set of 6,856 images and a test set of 2,940 images, ensuring a balanced and comprehensive evaluation. Training was conducted over 100 epochs, with the input image size fixed at 640

640 pixels. Standard data augmentation techniques were applied to enhance model robustness. The Ultralytics library was utilized for model implementation and training, as it supports the versions of YOLO evaluated in this study. The best-performing model from each training iteration was saved for subsequent evaluation.

This approach ensures that all models were assessed under uniform conditions, facilitating a fair and unbiased performance comparison.

3.4. Depth Estimation Module

Depth estimation is a critical component in autonomous firefighting drones, enabling the 3D localization of fire, smoke, and human targets. This spatial information is essential for precise water cannon targeting, enforcing safety constraints by avoiding suppression near detected humans, and supporting scene exploration when fire is not directly visible.

While stereo vision is a common approach for depth estimation, it is not well-suited for our RGB–thermal camera configuration due to the lack of shared visual features across the two modalities. Thermal images lack the distinct textures and edges found in RGB, making stereo correspondence unreliable. Although geometric calibration between the cameras was achieved, disparity-based depth estimation was inconsistent and unsuitable for real-time applications.

As a result, monocular depth estimation was adopted as a more practical and robust solution. It leverages deep learning models to infer depth from a single RGB image, eliminating the need for feature correspondence. The following subsections present a comparative evaluation of recent monocular depth models and justify the selected approach based on accuracy, speed, and system compatibility.

3.4.1. Monocular Depth Estimation

Monocular depth estimation refers to the process of inferring depth information from a single RGB image. This approach is particularly valuable in scenarios where multi-camera systems or stereo matching are impractical due to hardware constraints or lack of

visual feature correspondence. Recent advances in deep learning have enabled significant improvements in single-image depth estimation, with models now trained on large and diverse datasets to generalize across varied environments.

Depth estimation models can broadly be classified based on their architectural backbones (e.g., convolutional or transformer-based), loss functions (e.g., scale-invariant or affine-invariant), and output formats (e.g., relative vs. metric depth). In this study, we evaluate several state-of-the-art monocular depth models for their suitability in real-time UAV-based fire detection systems, focusing on their speed, accuracy, scalability, and ease of integration.

DepthAnything

DepthAnything is a scalable depth estimation framework that leverages a data engine designed to automatically collect and annotate large-scale unlabeled datasets. The model is built on a DINOv2 encoder for feature extraction and a DPT decoder for depth regression. It introduces two key training strategies: challenging data augmentations that enhance model generalization and auxiliary supervision that incorporates semantic priors. An affine-invariant loss is used to make the model robust to scale and shift inconsistencies. Its efficient architecture and high accuracy make it suitable for UAV-based perception tasks in variable environments.

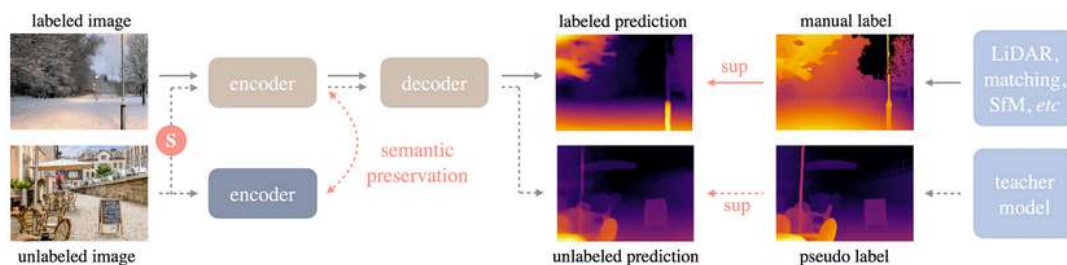


Figure 3.18: Depth Anything Framework

DepthCrafter

DepthCrafter is designed to improve fine-grained depth estimation using a multi-scale transformer encoder-decoder structure. It enhances local detail recovery through progressive upsampling and combines features from various spatial resolutions. The architecture is optimized for performance on high-resolution images and is effective in scenarios where object boundaries and spatial precision are critical. Although computationally heavier than some alternatives, DepthCrafter offers superior performance in complex environments.

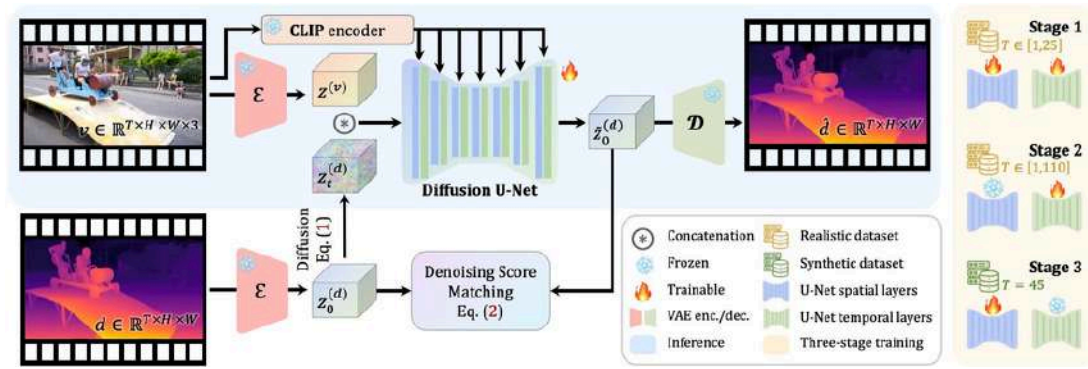


Figure 3.19: DepthCrafter Framework

Depth Pro

Depth Pro adopts a lightweight design that balances accuracy and inference speed. It utilizes a hybrid encoder combining convolutional and attention mechanisms, offering an efficient trade-off for real-time applications. Its architecture is optimized for embedded systems with limited computational resources, making it well-suited for deployment on resource-constrained drones. Despite its compact size, Depth Pro achieves competitive depth prediction performance in indoor and outdoor environments.

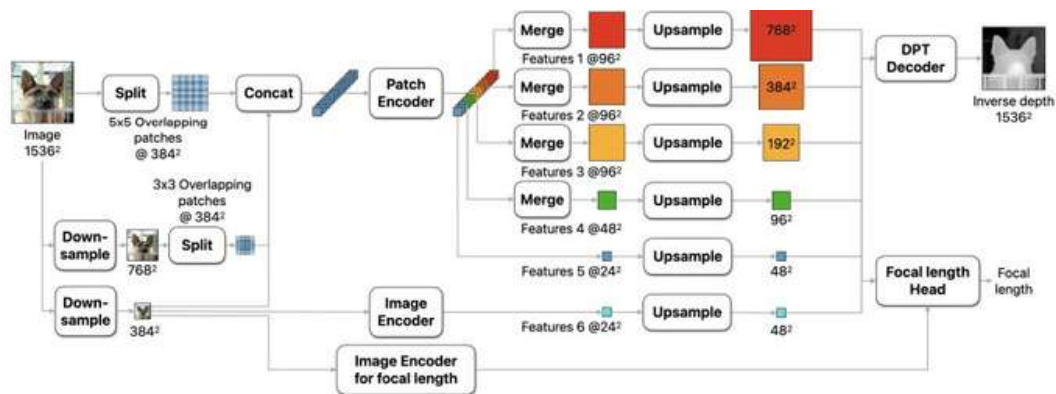


Figure 3.20: Depth Pro Architecture

Marigold

Marigold introduces a multi-task learning approach that jointly optimizes depth and surface normal estimation. This synergy allows the model to capture fine-grained geometric cues and improve generalization. It uses transformer-based layers for long-range context modeling and includes an uncertainty-aware loss function to handle noisy data. Marigold is particularly effective in environments with occlusions or ambiguous textures, although it may require higher computational resources.

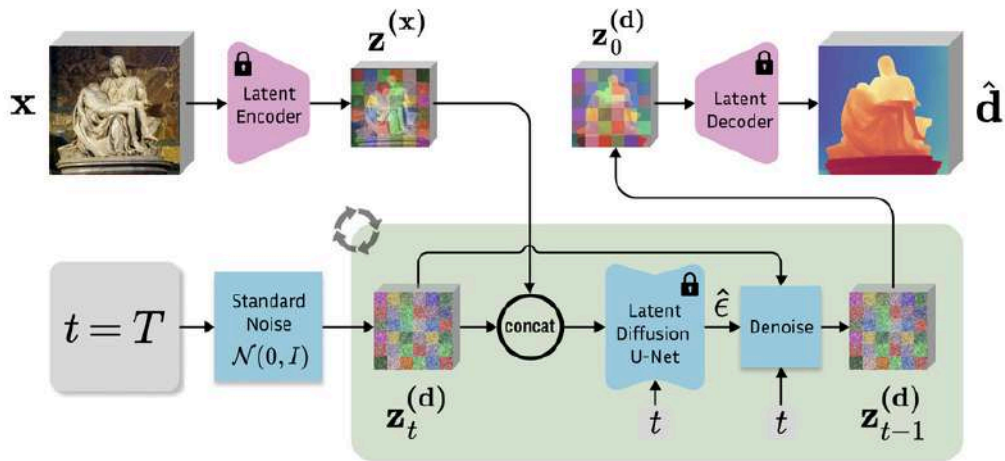


Figure 3.21: Marigold Framework

Metric3D

Metric3D focuses on improving metric depth estimation by introducing a canonical camera space transformation module that reduces scale ambiguity. Built upon a U-Net architecture with a DINOv2 backbone, it employs a novel random proposal normalization loss to ensure stable training across a massive dataset of over 8 million images. This model is designed to bridge the gap between relative and absolute depth prediction, which is essential for fire suppression systems requiring precise 3D localization.

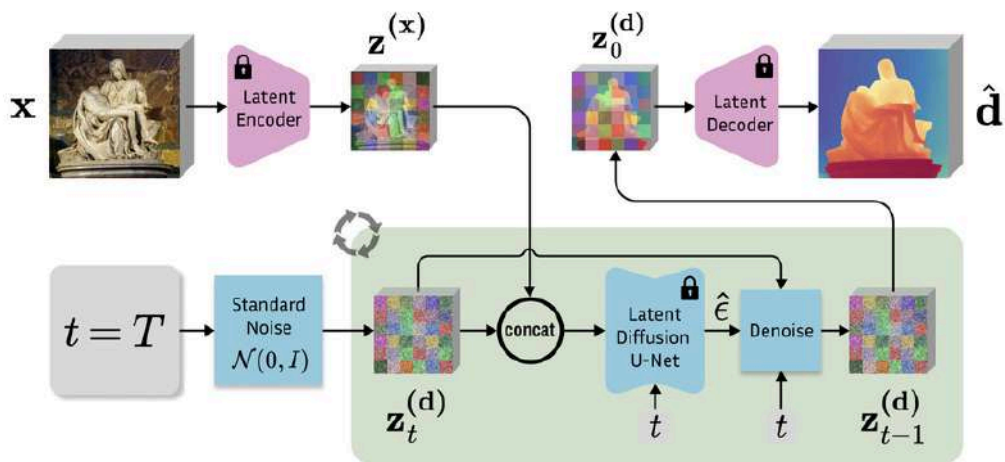


Figure 3.22: Metric3D Framework

MiDaS

MiDaS addresses dataset diversity and scale inconsistencies by using 3D movie data and stereo pairs to train on relative depth. It employs a multi-objective optimization strategy, treating each dataset as an independent task to avoid overfitting to a single domain. A scale- and bias-insensitive loss is used to reconcile discrepancies between datasets. The model's ResNet-based multi-scale architecture makes it well-suited for general-purpose depth estimation, providing strong relative depth cues even in complex environments.

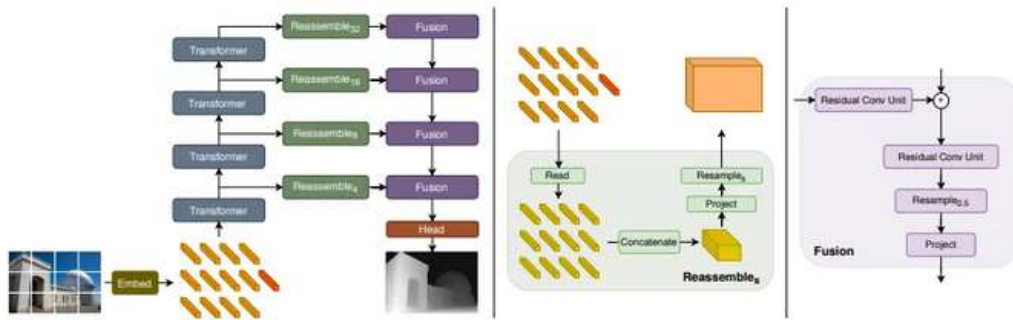


Figure 3.23: DPT Model Architecture

Table 3.10: Summary of state-of-the-art monocular depth estimation models.

Model	Training Datasets	Dataset Type	Model Resolution
MiDaS	DIML, MegaDepth, ReDWeb, 3DMovies	Indoor; Synthetic	38 × 38
DPT	DIML, WSVD, TartanAir, IRS, MegaDepth	Indoor; Driving; Synthetic	4 × 4
Metric3D	DDAD, DS, Argoverse2, Cityscapes, UASOL	Driving; Outdoor	38 × 38
ZoeDepth	KITTI, NYUDepth, MegaDepth, IRS	Indoor; Driving; Synthetic	4 × 4
DepthAnything	Objects365, SUN, ImageNet-21K, SA-1B	Mixed; General Purpose	51 × 96
DepthCrafter	KITTI, NYU, TartanAir	Indoor; Driving; Synthetic	2 × 0
DepthPro	ScanNet, KITTI, SceneFlow	Indoor; Driving; Synthetic	38 × 51
Marigold	Taskonomy, MegaDepth, ScanNet	Indoor; Synthetic	4 × 2

3.4.2. Existing Benchmark Datasets

Benchmark datasets are fundamental to the training and evaluation of monocular depth estimation models, as they provide standardized, diverse, and annotated data crucial for model generalization. These datasets are typically collected using either active sensors (e.g., RGB-D cameras, LiDAR) or passive stereo methods. While RGB-D sensors like Microsoft Kinect and Intel RealSense are easy to use and produce aligned data, their

effective range is limited. In contrast, LiDAR-based datasets offer high-resolution depth over longer distances, making them ideal for autonomous driving scenarios.

The KITTI dataset is one of the most widely used benchmarks, offering synchronized RGB and LiDAR data across diverse road scenes. NuScenes extends KITTI with more extensive annotations and sensor modalities. For indoor environments, NYU Depth provides RGB-D video sequences captured with structured-light sensors. Make3D focuses on outdoor landscapes, offering paired RGB and depth images.

Cityscapes, although primarily designed for semantic segmentation, includes stereo sequences useful for unsupervised training. Other datasets like DIODE and SceneFlow offer detailed geometry and synthetic stereo data, respectively, while MegaDepth leverages internet photo collections and multi-view reconstruction to generate relative depth.

Together, these datasets cover a wide range of use cases and environmental conditions, enabling the development of robust depth estimation methods. All the above datasets are summarized in Table 3.11, highlighting their applicable scenarios, acquisition tools, image resolutions, and dataset sizes.

Table 3.11: Summary of common benchmark datasets for depth estimation.

Name	Applicable Scenarios	Acquisition Tools	ImageResolution	NumberOfImages
KITTI	Autonomousdriving	LiDAR RGB-D	1242 × 375	93,000+
NYUDepth	Indoor Urban/city	Manualannotation	640 × 480	1,449
Cityscapes	Outdoorlandscapes	LiDAR LiDAR Faro	2048 × 1024	22,973
Make3D	Autonomousdriving	Rendered	2272 × 1704	534
NuScenes	Indoor/outdoorsynthesis	SFM+MVS RGB-D	1600 × 900	1,000
DIODE	Synthetic(stereo)		1024 × 768	27,858+
SceneFlow	Diversearchitecture		960 × 540	39,000+
MegaDepth	Forestenvironments		1600 × 1600	130,000+
LANet			1280 × 720	400

3.5. Safety and Targeting Decision-Making Logic

The decision-making system in autonomous firefighting plays a central role in linking perception outputs—such as fire, smoke, and human detections, along with depth estimation—to safe and effective suppression actions. Its purpose is to to extract valid fire source 3D point and determine whether the detected source is actionable, while ensuring no harm to humans in the scene. This logic incorporates spatial awareness, safety protocols, and

task prioritization to guide the targeting system. It uses detection results and depth cues to compute candidate fire locations and filters them based on human presence, distance thresholds, and classification confidence. This section presents the safety constraints applied, the target selection logic, and the system's ROS2-based implementation for real-time operation.

3.5.1. Safety Rules and Constraints

To ensure safe and reliable operation of the autonomous firefighting system, a set of safety rules and logic constraints is enforced before any suppression action is initiated. These rules are designed to prevent accidental harm to humans, avoid acting on uncertain data, and maintain robust decision-making under variable conditions.

The first and most critical rule is the enforcement of a ****safety buffer zone**** around detected humans. Specifically, no fire suppression action is permitted if a fire source point lies within a predefined minimum distance—typically set to 1.5—from any human center point. This spatial constraint is computed using the 2D Euclidean distance between the fire source point (projected from 3D) and the center coordinates of the detected human bounding box in the image frame:

$$d = \sqrt{(x_f - x_h)^2 + (y_f - y_h)^2}$$

where (x_f, y_f) is the fire source point and (x_h, y_h) is the human center point.

Additionally, fire detections with low classification confidence or unreliable depth estimations (e.g., missing or invalid depth values) are filtered out. This ensures the system only reacts to high-certainty observations, minimizing the risk of false alarms or misdirected suppression actions.

By combining geometric reasoning with detection confidence thresholds, the safety logic ensures that no fire suppression event occurs unless the target is well localized, verified by depth, and safely distanced from any detected humans.

3.5.2. Fire Source Point Estimation

To precisely localize the origin of a detected fire and ensure accurate targeting, this system employs a lightweight and robust classical approach that avoids reliance on computationally expensive segmentation models. The process begins by applying an adaptive HSV-based color mask to the input image to isolate fire-like pixel regions. This mask dynamically adjusts to lighting variations and scene content, enhancing robustness across diverse environments.

Following initial masking, morphological preprocessing is performed to refine the fire regions. Specifically, small isolated blobs are removed, and morphological closing operations are applied to fill gaps and connect fragmented regions. This step significantly improves the reliability of the subsequent source estimation. Once a coherent fire region is identified, two estimation strategies are employed depending on the availability of thermal information:

- With Thermal Data: The fire source point is identified as the pixel within the detected fire region (typically confined by the object detector’s bounding box) that exhibits the highest temperature or thermal intensity. This strategy leverages the high emissivity of active combustion zones, providing an accurate and physics-consistent localization of the ignition point.
- Without Thermal Data: When only RGB imagery is available, a weighted centroid approach is applied within the bottom portion of the detected fire blob. This region is selected based on the physical behavior of fire propagation, which typically rises from a base ignition zone. Pixel-wise weights are computed using a combination of vertical position (favoring pixels closer to the bottom of the region), brightness (favoring hotter-appearing pixels), and hue-based color temperature (favoring yellow pixels over cooler orange hues). This strategy ensures that the estimated source point lies within the most likely active combustion zone—near the base and within the hottest part of the fire—while being robust to flame shape, camera angle, and environmental noise.

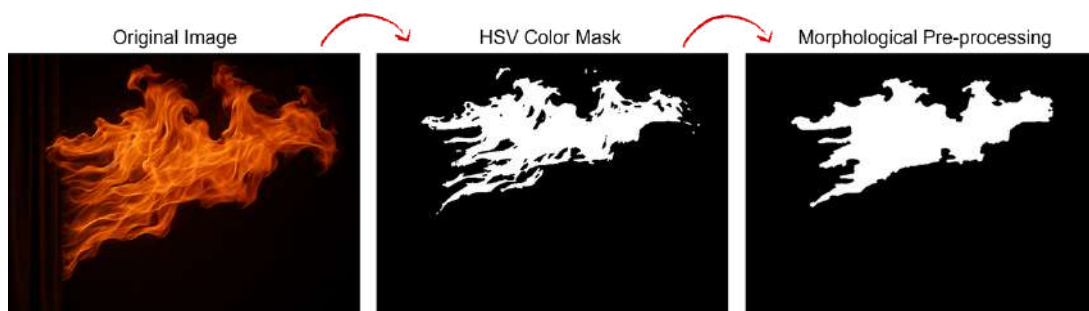


Figure 3.24: Classical fire masking pipeline

After determining the fire source pixel in the image plane, it is then projected into 3D space using the corresponding monocular depth estimation. The resulting 3D coordinate forms the basis for both safety checks and cannon targeting logic.

3.5.3. ROS2 Integration and Targeting Decision Process

The targeting and suppression logic is implemented as a dedicated ROS2 node integrated into the autonomous drone system. This node is responsible for selecting valid fire targets based on perception data, enforcing safety constraints, and issuing commands to the fire suppression mechanism.

Data Publishing Architecture: The perception node serves as the upstream data source. It publishes a unified message containing a list of detected objects, where each object includes:

- Object class (e.g., fire, human, smoke),
- 3D position estimated using monocular depth,
- Detection confidence,
- Bounding box coordinates,
- Safety flags (e.g., proximity to humans).

Targeting Decision Node: This node subscribes to the published perception message and processes it in real time. For each object labeled as fire, it evaluates:

1. Whether valid depth data is available for localization,
2. Whether the detection is sufficiently confident,
3. Whether the estimated fire position is outside the defined safety buffer zone around any detected human.

Output: For each valid fire detection, the node outputs:

- A 3D coordinate representing the fire source point, and
- A boolean flag `safe_to_suppress` indicating that the suppression system may safely engage.

If multiple fires meet all criteria, the nearest valid fire (based on Euclidean distance from the drone) is selected for targeting. This information is sent to both the path planning module and the cannon control system, enabling autonomous, responsive, and safe firefighting decisions.

Fallbacks and Error Handling: In the event of missing depth data, the system marks the fire instance as `unsafe_to_suppress`. When multiple humans are detected near a fire source, the safety radius is dynamically increased to ensure a wider buffer zone. Additionally, detections with low confidence are ignored by the decision logic to avoid false targeting or unnecessary actuation.

4. Results & Discussion

This chapter presents and analyzes the performance of the proposed autonomous perception and targeting system. It begins by evaluating the detection module across multiple state-of-the-art models, followed by an assessment of RGB-T fusion for improved perception. The effectiveness of monocular depth estimation methods under practical constraints is also explored. Furthermore, the system's safety logic and decision-making behavior are examined through real-world scenarios and performance metrics. Each component is discussed in terms of accuracy, robustness, and real-time applicability, offering insights into system strengths and operational limitations.

4.1. Detection Module

Accurate and efficient detection of fire, smoke, and human presence is essential for autonomous firefighting. To address this, a range of object detection models—spanning multiple YOLO versions and architectural scales—were trained and evaluated. The goal is to identify models that strike the best balance between detection reliability and real-time performance on embedded UAV platforms. The evaluation considers precision, recall, mean average precision (mAP), and inference speed. Results are grouped into two categories: lightweight models optimized for deployment under strict resource constraints, and larger or transformer-based models capable of higher accuracy at the cost of increased computational demands.

4.1.1. Evaluation of Lightweight Detection Models (Nano and Small)

This subsection analyzes the performance of lightweight models—specifically the nano and small variants of YOLOv5 through YOLOv12—selected for their potential real-time viability on embedded UAV platforms. These models are of particular interest due to their balance of efficiency and accuracy within computational constraints.

Figure 4.2 presents the training convergence of these models in terms of mAP@0.5:0.95 across epochs. Among the nano models, YOLOv12n consistently outperformed earlier variants, reaching a peak mAP of 0.6638, followed closely by YOLOv9t and YOLOv10n.

This improvement is attributed to architectural innovations such as optimized attention mechanisms and improved feature aggregation. However, it is also observed that these models occasionally suffer from spatial inconsistencies—such as overlapping or duplicate bounding boxes—particularly in dense scenes or under partial occlusion 4.1. This behavior may stem from limited feature resolution and anchor box precision, which is more pronounced in nano variants due to their constrained depth and width.



Figure 4.1: Overlapping and duplicate fire detections for lightweight YOLO models

In contrast, the small models showed a notable leap in performance. YOLOv12s achieved the highest mAP among all small variants at 0.7161, along with strong fire precision (0.988) and recall (0.949). YOLOv11s and YOLOv8s also maintained high fire detection precision above 0.98 and stable recall above 0.94. These models offer a compelling trade-off, achieving near state-of-the-art accuracy while remaining suitable for deployment on moderate embedded platforms. Their improved backbone structures and decoupled detection heads contribute to more stable bounding box predictions, reducing localization errors observed in nano models.

To further assess robustness, confidence threshold sweeps were performed for all small models (see Figures 4.3). This analysis illustrates how each model's precision, recall, and F1 score vary as the confidence threshold increases. Across all models, F1 scores tend to plateau up to thresholds around 0.5–0.6 before declining, suggesting this range is optimal for operational deployment. YOLOv12s and YOLOv11s exhibit consistently high precision and recall across a broad range of confidence thresholds, indicating a stable response under varying detection confidence requirements. In contrast, YOLOv8s shows a more noticeable trade-off between precision and recall, where precision increases at high thresholds but recall declines more steeply. This behavior implies that YOLOv8s adopts a more conservative prediction strategy as confidence filters tighten. Meanwhile, YOLOv5s, being the oldest architecture among them, exhibits a less consistent F1 profile, with early degradation in both precision and recall as the threshold increases—highlighting its limited adaptability in dynamic decision scenarios. Toward the extreme end of the confidence

Table 4.1: Results for Nano YOLO Models on the Fire, Smoke, and Human dataset.

Mo del	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (FPS)	Inference Time (ms)
YOLOv5n	Fire	0.92	0.92	0.86	0.69	357.1	2.8
	Smoke	1	2	0	9		
	Human	0.78	0.77	0.72	0.59		
	All	5	8	9	3		
YOLOv8n	Fire	0.69	0.57	0.58	0.47	322.6	3.1
	Smoke	0	0	2	3		
	Human	0.79	0.75	0.72	0.58		
	All	9	7	4	8		
YOLOv9t	Fire	0.89	0.93	0.88	0.71	238.1	4.2
	Smoke	6	1	4	9		
	Human	0.78	0.77	0.75	0.61		
	All	9	4	6	5		
YOLOv10n	Fire	0.72	0.57	0.62	0.50	476.2	2.1
	Smoke	1	8	1	5		
	Human	0.80	0.76	0.75	0.61		
	All	2	1	4	3		
YOLOv11n	Fire	0.91	0.92	0.93	0.75	370.4	2.7
	Smoke	4	6	2	8		
	Human	0.78	0.78	0.79	0.64		
	All	2	8	4	6		
YOLOv12n	Fire	0.71	0.58	0.64	0.52	400.0	2.5
	Smoke	0	4	9	8		
	Human	0.80	0.76	0.79	0.64		
	All	2	5	2	4		
	Fire	0.89	0.92	0.90	0.73		
	Smoke	7	7	8	8		
	Human	0.78	0.73	0.75	0.61		
	All	0.73	0.58	0.64	0.52		
	Fire	0.80	0.75	0.77	0.62		
	Smoke	3	5	8	7		
	Human	0.87	0.92	0.90	0.73		
	All	0.79	0.79	0.80	0.65		
	Fire	0.73	0.55	0.63	0.51		
	Smoke	4	9	9	7		
	Human	0.79	0.80	0.82	0.67		
	All	1	4	4	0		
	Fire	0.72	0.56	0.65	0.53		
	Smoke	0.94	0.92	0.96	0.78		
	Human	4	9	9	7		
	All	0.79	0.80	0.82	0.67		
	Fire	0.82	0.76	0.81	0.66		
	Smoke	6	4	7	4		
	Human	0.82	0.76	0.81	0.66		
	All	0	6	6	4		

range (near 1.0), all models suffer rapid degradation due to overly strict filtering, which excludes even correct detections.

These confidence-based behaviors are particularly important in real-world deployments, where operating points must balance false positives with missed detections. Maintaining the F1-optimal threshold between 0.5–0.7 ensures robust fire detection while mitigating unnecessary cannon activations or missed suppression opportunities.

In summary, while nano models offer clear advantages in speed and memory footprint, small models—especially YOLOv12s and YOLOv11s—provide a reliable and balanced solution for accurate fire localization, vital to 3D targeting and autonomous decision-making.

Table 4.2: Results for Small YOLO Models on the Fire, Smoke, and Human dataset.

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (FPS)	Inference Time (ms)
YOLOv5s	Fire	0.73	0.88	0.90	0.70	212.8	4.7
	Smoke	1	0	0	0		
	Human	0.73	0.69	0.80	0.62		
	All	2	0	0	3		
YOLOv8s	Fire	0.57	0.67	0.69	0.54	188.7	5.3
	Smoke	3	0	5	1		
	Human	0.67	0.74	0.79	0.62		
	All	8	7	8	1		
YOLOv9s	Fire	0.98	0.94	0.99	0.81	135.1	7.4
	Smoke						
	Human	6	5	1	2		
	All	0.81	0.84	0.85	0.69		
YOLOv10s	Fire	0.76	0.54	0.65	0.53	270.3	3.7
	Smoke						
	Human	9	3	3	5		
	All	0.85	0.77	0.83	0.68		
YOLOv11s	Fire	0.86	0.93	0.96	0.78	243.9	4.1
	Smoke						
	Human	5	0	8	7		
	All	0.78	0.78	0.84	0.69		
YOLOv12s	Fire	0.68	0.58	0.68	0.55	217.4	4.6
	Smoke						
	Human	7	6	3	5		
	All	0.77	0.76	0.83	0.67		

4.1.2. Evaluation of Medium and Transformer-Based Models

For scenarios where computational resources are less constrained, medium-sized and transformer-based models offer a compelling advantage in terms of detection performance and robustness. This subsection evaluates YOLOv12m as a representative medium-scale CNN model, and two transformer-based architectures—RT-DETRv2-m and RF-DETR-B. Figure 4.4 shows the convergence behavior of the three models. RF-DETR-B demonstrates the fastest convergence during training, reflecting its design optimization for fine-tuning with fewer epochs. While YOLOv12m and RT-DETRv2-m gradually close the performance gap, RF-DETR-B consistently maintains a slight lead in terms of mAP@0.5:0.95, reaching 0.7452 versus 0.7433 for YOLOv12m and 0.7407 for RT-DETRv2-m.

As summarized in Table 4.3, transformer-based models provide marginally higher accuracy, particularly in object localization precision. However, this comes with a trade-off

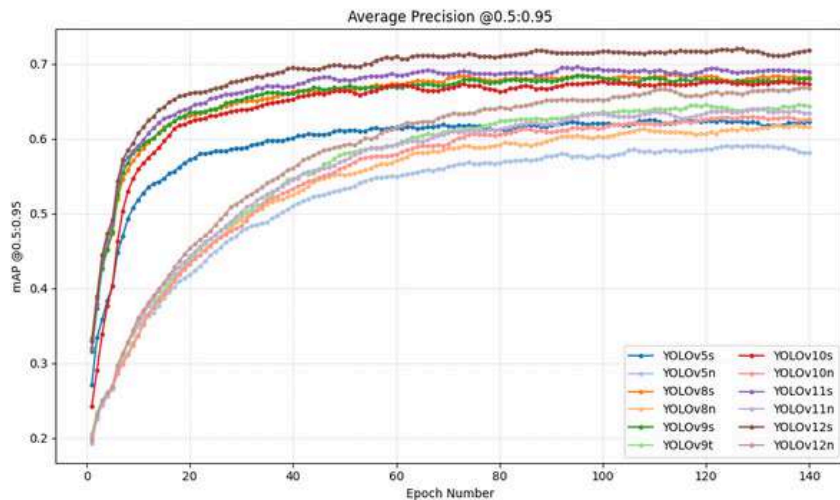


Figure 4.2: Training convergence (mAP@0.5:0.95) for nano and small YOLO variants.

in inference speed. Despite being capable of real-time operation, RF-DETR-B runs at 106.7 FPS and RT-DETRv2-m at 85.3 FPS—both notably slower than YOLOv12m, which maintains 131.7 FPS. This latency difference can be critical in time-sensitive robotic applications such as aerial firefighting.

When balancing detection quality and real-time responsiveness, YOLOv12m remains a favorable choice. It achieves comparable accuracy to transformer models while maintaining superior speed and lower computational overhead, making it well-suited for embedded systems on UAV platforms.

Table 4.3: Performance comparison of medium and transformer-based detection models.

Model	Class	Precision	Recall	mAP@50	mAP@50:95	Speed (FPS)	Inference Time (ms)
YOLOv12m	Fire	0.99	0.99	0.99	0.87	131.8	7.6
	Smoke	0	0	0	2		
	Human	0.90	0.89	0.92	0.75		
	All	2	6	8	3		
	Fire	0.81	0.65	0.74	0.60		
RT-DETRv2-m	Smoke	0	6	6	5	85.3	11.7
	Human	0.90	0.84	0.91	0.74		
	All	0	7	6	3		
	Fire	0.98	0.98	0.98	0.86		
	Smoke	7	7	7	9		
RF-DETR-B	Human	0.89	0.89	0.92	0.75	106.7	9.4
	All	0.80	0.65	0.74	0.60		
	Fire	7	4	3	3		
	Smoke	0.89	0.84	0.91	0.74		
	Human	0.99	0.99	0.99	0.87		
	Fire	0	0	0	4		
	Smoke	0.90	0.89	0.93	0.75		
	Human	0.81	0.65	0.74	0.60		
	All						
	Fire						

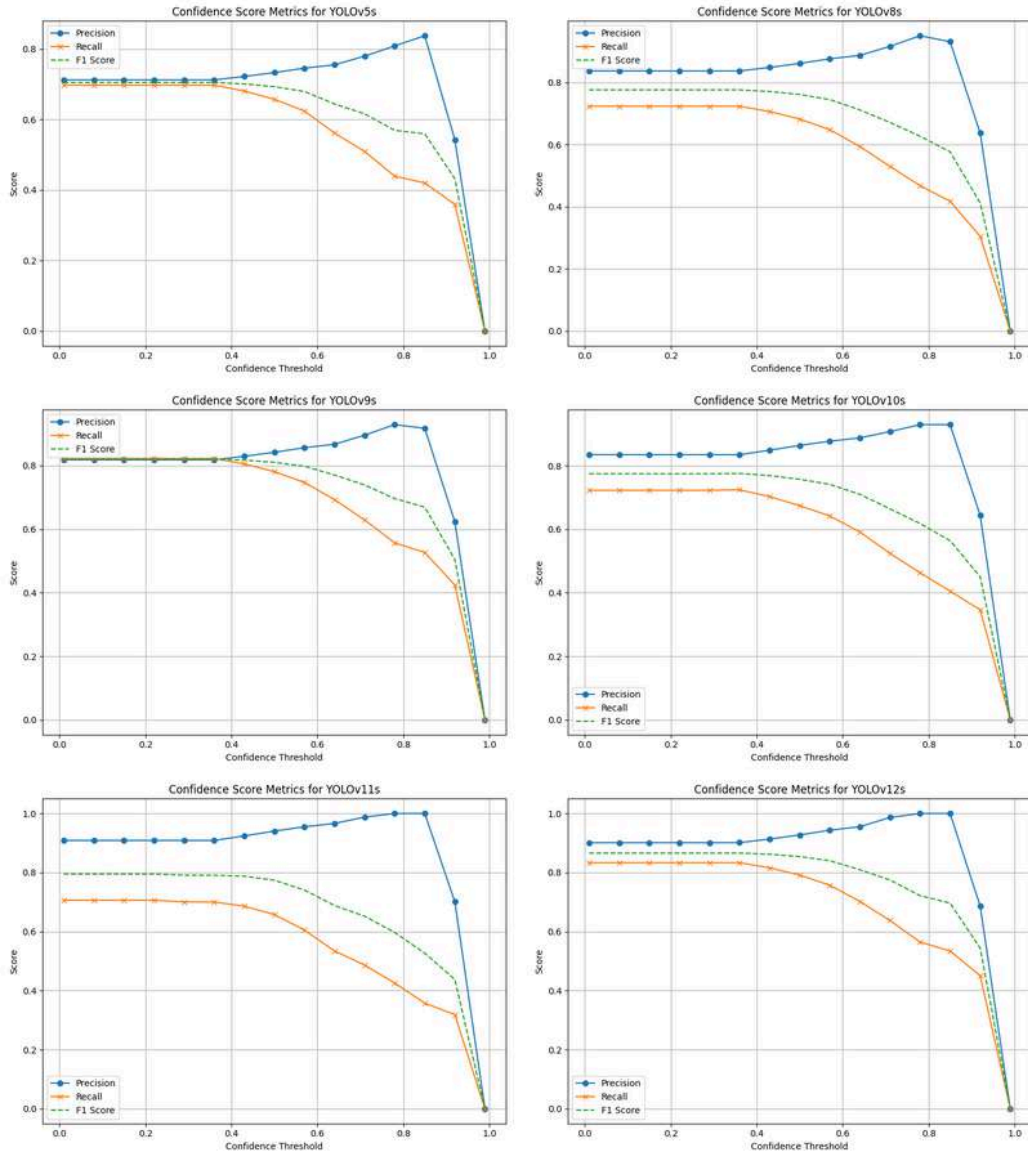


Figure 4.3: Precision, Recall, and F1 score as a function of confidence threshold for small YOLO variants.

4.1.3. Fusion of Thermal and RGB Imagery

To evaluate the benefits of combining thermal and RGB data, this section presents both the calibration results for precise image alignment and a comparative analysis of detection performance across single modalities and their fusion. First, we assess the accuracy of our multi-spectral stereo calibration, which enables pixel-level registration between thermal and RGB images. Then, we compare detection results using RGB-only, thermal-only, and fused RGB-T input pipelines. The goal is to quantify the added value of fusion in terms of detection robustness under challenging conditions such as smoke, lighting variations, and partial occlusion.

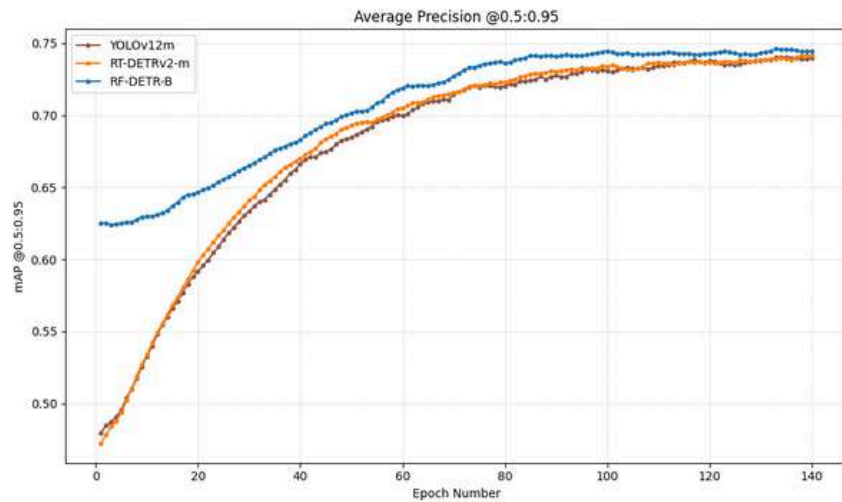


Figure 4.4: Training convergence (mAP@0.5:0.95) of YOLOv12m vs transformer-based models.

Multi-spectral Stereo Calibration

To enable pixel-level fusion of thermal and RGB data, a stereo calibration was performed between the two sensors using a custom multi-spectral checkerboard. Figure 4.5 visualizes the extrinsic calibration, showing the estimated poses of the calibration board from both camera perspectives. This calibration is critical for aligning thermal and RGB data for accurate fused detection.

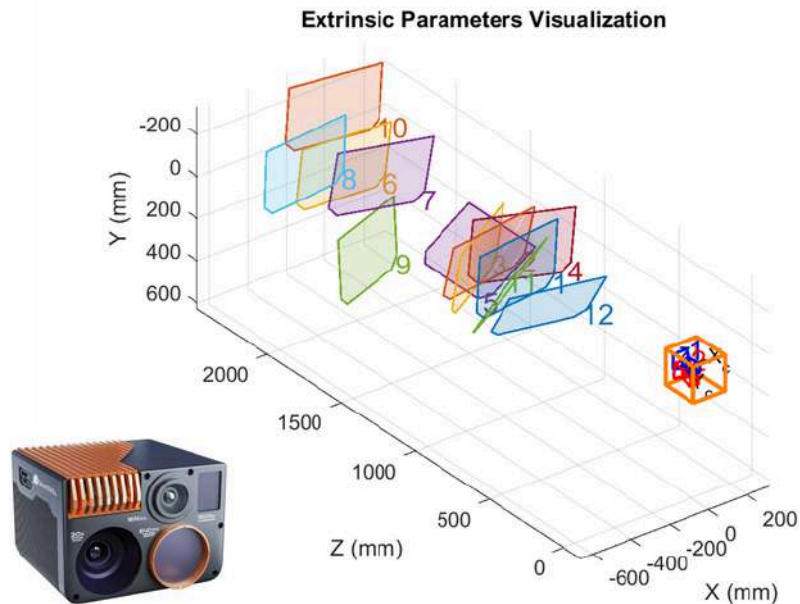


Figure 4.5: Extrinsic calibration visualization showing relative camera poses.

The intrinsic matrices for the RGB and thermal cameras are expressed in the standard

pinhole camera model format:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

The calibrated intrinsics are:

RGB Camera Intrinsics:

$$K_{RGB} = \begin{bmatrix} 1989.31 & 0 & 317.25 \\ 0 & 1502.93 & 317.25 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

Thermal Camera Intrinsics:

$$K_{Thermal} = \begin{bmatrix} 509.880 & 384.97 & 511.75 \\ 0 & 229.54 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

The stereo transformation from thermal to RGB frame is defined by a rotation matrix R and a translation vector T :

$$R = \begin{bmatrix} 0.9938 & 0.0254 & 0.1084 \\ 0.0285 & 0.99 & -0.0275 \\ -0.1076 & 0.9937 & 0.00304 \end{bmatrix} \quad T = \begin{bmatrix} 9.71 \\ 40.03 \\ 15.87 \end{bmatrix} \quad (4.4)$$

These calibration parameters were used to rectify and align RGB and thermal imagery, enabling pixel-wise fusion and depth-informed overlay generation. Uncertainties in the intrinsic estimates were captured during calibration and are provided in the supplementary distortion table (Table 4.4).

Table 4.4: Radial Distortion Coefficients and Uncertainties

Camera	Distortion Coefficients	Mean	Uncertainty (±)
RGB	k_1	0.014	0.009
	k_2	6	9
Thermal	k_1	-0.051	0.047
	k_2	0.5270	0.1308

Evaluation of RGB-T Fusion and Single Modalities

To assess the benefits of multi-spectral data fusion, we trained three instances of the YOLOv12s model using separate datasets: one based solely on RGB images, another on thermal images, and a third on aligned RGT fused inputs. The goal is to quantify the impact of each modality—and their combination—on detection performance, particularly for fire detection under diverse environmental conditions.

Table 4.5 summarizes the performance of each model using standard detection metrics: precision, recall, mAP@50, and mAP@50:95. The RGT model outperformed both single-modality models across all metrics, demonstrating strong generalization, reduced false positives, and robust detection in edge cases.

Table 4.5: Performance of YOLOv12s trained on different datasets for fire detection.

Dataset	Precision	Recall	mAP@50	mAP@50:95
RGT	0.87	0.98	0.97	0.61
RGB	0.85	0.7	0.8	0.8
Thermal	0.85	0.88	0.97	0.60

5

1

2

5

0.76

0.96

0.93

0.48

7

9

4

1

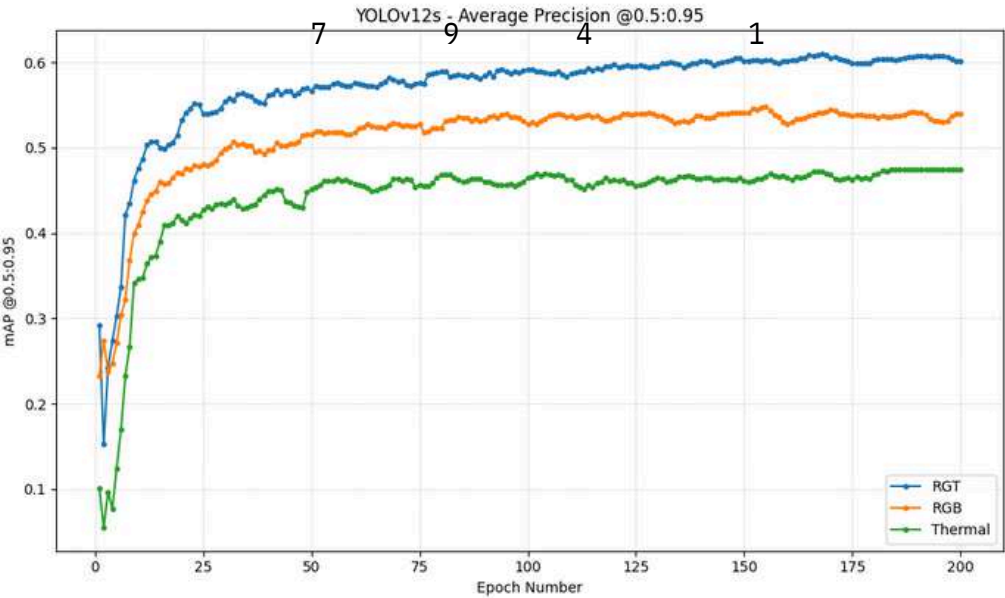


Figure 4.6: Training convergence (mAP@0.5:0.95) of YOLOv12s across different input modalities.

The RGB-based model achieved the high precision (0.855), indicating strong confidence in its fire predictions and a low rate of false positives. However, its recall (0.881) was slightly lower, suggesting it may fail to detect less visually prominent fires. As shown in

Figure 4.7, these missed detections often correspond to incipient or low-intensity flames that are hard to visually distinguish from the background, particularly during early ignition stages.

In contrast, the thermal-based model achieved the high recall (0.969) but suffered from lower precision (0.767), reflecting its sensitivity to heat signatures. This model tends to detect nearly all fire sources, including small or early-stage fires, but is prone to false alarms—such as confusing hot but benign objects with active flames. Figure 4.7 illustrates this effect, where high-temperature regions triggered incorrect fire detections.

The RGT model strikes the best balance with both the highest precision (0.873) and the highest recall (0.987), indicating it can reliably detect most fires while maintaining a low false positive rate. This balanced performance results in the highest mAP@50:95 of 0.618, making RGT fusion the most robust and generalizable approach, particularly in complex outdoor environments with variable lighting and thermal distractors. The learning curve in Figure 4.6 also highlights faster and more stable convergence when training with fused modalities.

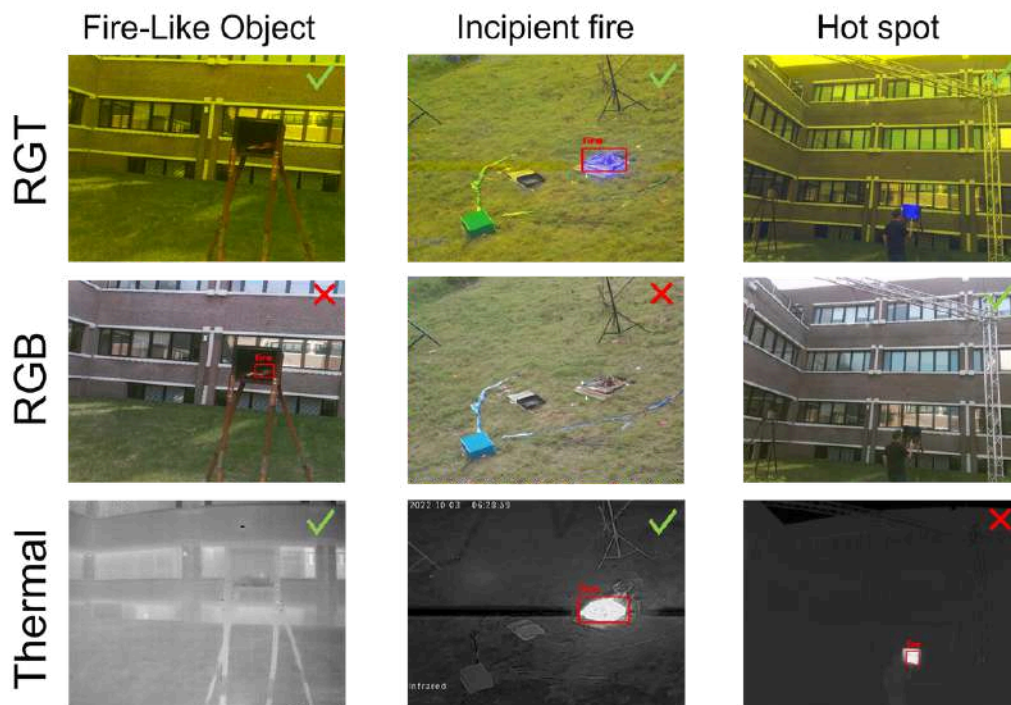


Figure 4.7: Example of missed fire by RGB-only model: incipient flames not visually prominent.

These results strongly support the integration of RGB and thermal modalities for fire detection in autonomous systems, enhancing robustness and ensuring greater operational safety and responsiveness.

4.2. Depth Estimation Results

Depth perception plays a critical role in enabling 3D localization of hazards and victims within the autonomous firefighting framework. This section evaluates the monocular depth estimation models integrated into the system, with a particular focus on their suitability for real-time performance in resource-constrained UAV deployments. The results are divided into two subsections: an architectural review of leading depth models, followed by an analysis of their conversion into usable metric depth outputs.

4.2.1. Monocular Depth Estimation Architectures

To assess single-frame depth inference, we evaluated a variety of state-of-the-art monocular depth estimation models—each representing different trade-offs between architectural complexity, speed, and depth accuracy. As summarized in Table 4.6, models such as Depth Anything V2, MiDaS 3.0, and Marigold exhibit strong generalization performance while maintaining lightweight configurations that make them viable for real-time inference. In particular, Depth Anything V2 stands out for its fine detail recovery and robust performance on reflective or transparent surfaces. Likewise, MiDaS 3.0 offers strong generalization to diverse environments, though it primarily produces relative depth outputs. Other models such as Metric3D and Depth Pro are designed to return metric depth, but often at the cost of increased computational demand or limited support for training on custom data.

From a task-specific evaluation standpoint, shown in Figure 4.8 Depth Anything V2 also showed superior robustness in estimating depth around challenging fire scenes—especially under thermal distortion or intense lighting. It was followed in effectiveness by Depth Pro, which performed well in structured environments but occasionally struggled with soft or irregular boundaries. DepthCrafter and MiDaS offered comparable performance, though both were more susceptible to local inconsistencies in homogeneous regions. Marigold, while fast and compact, showed the least stability around flickering or dynamic fire sources, making it less suitable for this domain-specific deployment.

Table 4.6: Comparison of popular monocular depth estimation models.

Model	Specialty	Speed/Size	Training	License
Depth Anything V2	Fine details, reflective & transparent/translucent surfaces	Small (99.2 MB), Base (390 MB), Large (1.34 GB)	Yes	Apache 2.0
DepthCrafter	Videos	3.05 GB	No	Tencent
DepthPro	Boundary, Metric, FocalLength	1.9 GB	No	Apple
Marigold	Scale-consistent depth estimation	167–335 MB	Yes	Apache 2.0
Metric3D	Metric	Small (150 MB), Large (1.65 GB)	Yes	BSD2
Midas3.0	Generalization across diverse scenes	Large (1.37 GB), Hybrid (490 MB)	Unofficial	MIT



Figure 4.8: Models understanding for fire scenes

4.2.2. Metric Depth Map Estimation

While most models optimized for real-time monocular depth estimation—such as Depth Anything V2, MiDaS, and Marigold—offer impressive speed and generalization, their outputs are inherently in relative scale. This presents a critical limitation for autonomous systems, particularly in firefighting drones, where accurate metric distance is essential for safe navigation, targeting, and actuation.

To address this, our system incorporates two strategies to convert relative depth maps into metric depth representations.

Lightweight Calibration-Based Scaling

The first method applies a lightweight post-processing step based on known geometric references or single-point range measurements. Two practical solutions are considered:

- **Laser-Based Single Point Scaling:** A compact laser module onboard the drone captures the actual distance to a reference point in the scene. This measured value is compared to the corresponding relative depth value from the predicted map, and a global scaling factor is computed and applied across the entire depth map.
- **Known Object Size (Water Cannon Tip):** Alternatively, when a known-sized object such as the water cannon tip (40 mm in diameter) is consistently visible in the scene, its projected size in pixels provides a means to estimate scale. This method leverages the camera’s intrinsic parameters and geometric principles to infer real-world scale without requiring additional sensors. In our experiments, this method achieved an average error of 24 cm compared to the ground truth

measurement at 5 meters, as shown in Figure 4.9.

These calibration techniques offer a practical, sensor-efficient solution to achieving metric scaling from relative maps, with minimal computational cost—making them ideal for resource-constrained embedded systems.



Figure 4.9: Calibration-based metric depth estimation

ZoeDepth: Learning-Based Metric Scaling

For a more generalized and model-agnostic solution, we integrate ZoeDepth, a learning-based framework specifically developed to infer metric depth from relative predictions. ZoeDepth augments base models, such as MiDaS, with a scale regression head that learns a global or depth-wise scaling factor. This module is trained on datasets with absolute depth labels (e.g., KITTI, NYU), enabling the system to map scale-ambiguous predictions to real-world distances.

In this study, we adopt Depth Anything V2 instead of MiDaS as a base model for ZoeDepth framework. This decision is motivated by Depth Anything V2's demonstrated robustness in detecting fine structures and high-temperature objects, such as fire regions, which are critical for the drone's safety logic and targeting accuracy. When applied in our setup, it achieved a lower average error of 16 cm on the same 5-meter target, as demonstrated in Figure 4.10. The combined system enables real-time, single-image depth estimation with outputs in physical metric units—suitable for 3D localization, point cloud generation, and interaction-aware actuation. While slightly more computationally intensive than direct calibration-based methods, the added generality and robustness make this approach suitable for field deployments where onboard sensing may be limited.

4.3. Safety and Targeting Decision-Making Logic

This section evaluates the system's ability to safely and accurately select suppression targets based on perception outputs. Key components include fire source localization and



Figure 4.10: ZoeDepth-based metric depth estimation

filtering logic to ensure that only safe and actionable targets are passed to the cannon control system. We analyze the performance of source point estimation with and without thermal data and assess the robustness of our lightweight masking approach in diverse fire scenarios.

4.3.1. Fire Source Point Targeting

When thermal data is available, identifying the fire source is straightforward: the system selects the pixel within the fire bounding box with the highest thermal intensity. This approach consistently aligns with the true ignition point, as shown in Figure 4.11.

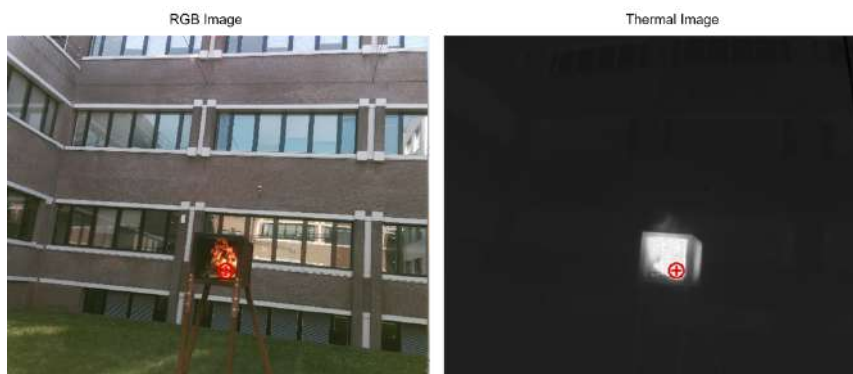


Figure 4.11: Thermal-based fire source localization

In RGB-only scenarios, early attempts to use the bounding box's bottom-center proved unreliable, particularly for side-originating fires. Figure 4.12 illustrates such a failure case.

To address this, we introduced an adaptive HSV mask combined with morphological filtering. This isolates coherent flame regions from which a weighted centroid is calculated, considering brightness and position cues. As shown in Figure 4.13, this classical method captures the fire fine details and performs comparably to segmentation models while requiring significantly less computation.

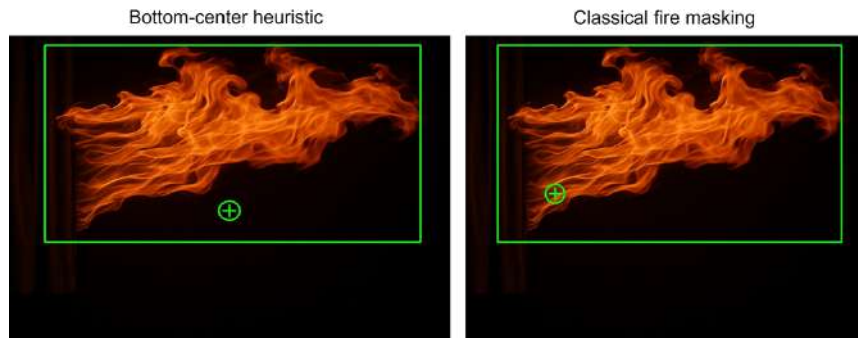


Figure 4.12: Bottom-center heuristic and classical fire masking for side-propagating fire.

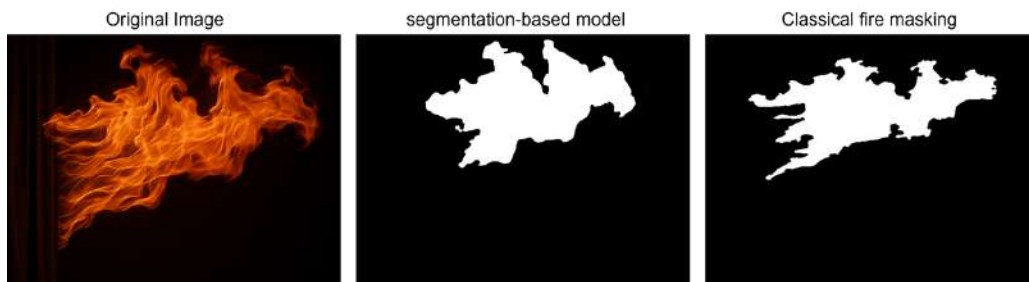


Figure 4.13: Comparison of classical fire masking and segmentation-based model.

4.3.2. Safety Rules and Constraints

To validate the effectiveness of the safety enforcement logic, we examined the system's decision-making behavior in scenarios involving nearby human presence. Figure 4.14 illustrates a real-world test case where two fire detections are evaluated. Although both fires are confidently identified, the system marks only one as `safe_to_suppress`, based on the proximity of a detected person.

The left fire, although valid in appearance, is within the predefined 1.5 safety buffer of a human and is thus blocked from suppression. In contrast, the right fire is safely distanced and flagged as actionable. This demonstrates the system's ability to filter risky targets and uphold spatial safety constraints in real time.



Figure 4.14: Safety rules and constraints logic

5. Conclusion and Future Work

This thesis presented the design, implementation, and evaluation of a vision-based Fire Detection and Ranging (FDAR) system tailored for autonomous firefighting drones. The proposed system integrates multi-modal perception, spatial localization, and decision-making logic within a modular ROS2 framework, enabling real-time detection and safe suppression of fire in complex and dynamic environments.

The detection pipeline demonstrated the benefits of combining RGB and thermal imagery, with RGB-T fusion outperforming single-modality models in both accuracy and robustness. Extensive benchmarking across multiple object detection architectures—including the YOLO family, RT-DETRv2, and RF-DETR—highlighted the trade-offs between model complexity, performance, and real-time suitability for UAV deployment. Additionally, monocular depth estimation was successfully integrated to provide 3D localization of detected targets, enabling spatial reasoning and safe cannon targeting.

A key innovation in the system is the fire source point estimation mechanism, which combines classical computer vision techniques and adaptive heuristics to localize ignition points without relying on computationally expensive segmentation models. Coupled with a human-aware safety logic, the system ensures suppression actions are only initiated when fire targets are confidently localized and no humans are present within hazardous proximity.

FutureWork

While the current system demonstrates strong performance in controlled and semi-structured outdoor environments, several opportunities exist for future enhancement:

- Integration with Autonomous Navigation: Future work should focus on coupling the FDAR module with a UAV flight controller and path-planning algorithms, enabling fully autonomous fire suppression missions from detection to actuation.
- Onboard Deployment and Optimization: Although inference speeds were benchmarked for real-time constraints, further optimization is needed for seamless deployment on lightweight edge hardware aboard UAVs. Techniques such as model

quantization and pruning may be explored.

- Advanced Multi-modal Fusion: Future iterations can explore deeper fusion strategies, such as feature-level or attention-based fusion, to more effectively combine RGB and thermal information beyond pixel alignment.
- Human State Tracking and Reporting: An important extension involves the development of a state-aware tracking module capable of estimating and reporting the number of humans within the fire scenario, along with their consciousness state (conscious vs. unconscious). This will require augmenting the dataset with annotated examples of unconscious individuals and adapting the detection model accordingly. Additionally, a robust identity-preserving tracker must be implemented to maintain consistent human IDs over time, ensuring accurate and reliable situation reporting for emergency responders.
- Temporal Reasoning and Event Detection: Incorporating video-based temporal models or early fire event prediction may improve detection stability and allow proactive suppression before fires grow uncontrollably.
- Expanded Environmental Testing: Evaluation in more diverse and unstructured environments—such as forested areas, industrial facilities, or nighttime conditions—will further validate system generalizability and robustness.

Overall, the proposed system serves as a promising step toward safer, faster, and more effective autonomous firefighting, laying the groundwork for future research in aerial robotic intervention and intelligent emergency response.

Bibliography

- [1] Y. Cao, F. Yang, Q. Tang, and X. Lu. An attention enhanced bidirectional lstm for early forest fire smoke recognition. *IEEE Access*, 7:154732–154742, 2019.
- [2] Y. Cao, Q. Tang, and X. Lu. Stcnet: Spatiotemporal cross network for industrial smoke detection. *Multimedia Tools and Applications*, 81(7):10261–10277, 2022.
- [3] T.-H. Chen, P.-H. Wu, and Y.-C. Chiou. An early fire-detection method based on image processing. In *International Conference on Image Processing*, volume 3, pages 1707–1710, 2004.
- [4] Yilun Chen, Zihan Liu, Haoyu Peng, Yue Gao, and Ming Sun. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2312.05810*, 2023.
- [5] M. Correa, G. Hermosilla, R. Verschae, et al. Human detection and identification by robots using thermal and visual information in domestic environments. *Journal of Intelligent Robotic Systems*, 66(1-2):223–243, 2011. doi: 10.1007/s10846-011-9563-1.
- [6] T. M. Dawdi, N. Abdalla, Y. M. Elkalyoubi, et al. Locating victims in hot environments using combined thermal and optical imaging. *Computers and Electrical Engineering*, 85:106697, 2020. doi: 10.1016/j.compeleceng.2020.106697.
- [7] P. V. A. de Venâncio, R. J. Campos, T. M. Rezende, A. C. Lisboa, and A. V. Barbosa. A hybrid method for fire detection based on spatial and temporal patterns. *Neural Computing and Applications*, 35(13):9349–9361, 2023.
- [8] A. Filonenko, D. C. Hernández, and K.-H. Jo. Fast smoke detection for video surveillance using cuda. *IEEE Transactions on Industrial Informatics*, 14(2):725–733, 2017.
- [9] A. Filonenko, L. Kurnianggoro, and K.-H. Jo. Smoke detection on video sequences using convolutional and recurrent neural networks. In *Computational Collective Intelligence: 9th International Conference*, pages 558–566, 2017.

- [10] S. S. Ghidary, Y. Nakata, T. Takamori, et al. Human detection and localization at indoor environment by home robot. In 2000 IEEE International Conference on Systems, Man, and Cybernetics, pages 1360–1365. IEEE, 2000. doi: 10.1109/ICSMC.2000.886043.
- [11] J. Han and B. Bhanu. Fusion of color and infrared video for moving human detection. *Pattern Recognition*, 40(6):1771–1784, 2007. doi: 10.1016/j.patcog.2006.11.018.
- [12] C. L. Herry, M. Frize, and R. A. Goubran. Segmentation and landmark identification in infrared images of the human body. In 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, pages 957–960, New York City, NY, USA, 2006. IEEE. doi: 10.1109/IEMBS.2006.260077.
- [13] W.-B. Horng, J.-W. Peng, and C.-Y. Chen. A new image-based real-time flame detection method using color analysis. In *IEEE Networking, Sensing and Control*, pages 100–105, 2005.
- [14] J. Huang, Z. He, Y. Guan, and H. Zhang. Real-time forest fire detection by ensemble lightweight yolox-l and defogging method. *Sensors*, 23(4):1894, 2023.
- [15] M. Ivašćić-Kos, M. Kršćto, and M. Pobar. Human detection in thermal imaging using yolo. In 2019 Fifth International Conference on Computer and Technology Applications, pages 20–24, Istanbul, Turkey, 2019. Association for Computing Machinery. doi: 10.1145/3323933.3324076.
- [16] F. B. Jaradat and D. Valles. A victims detection approach for burning building sites using convolutional neural networks. In 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pages 280–286, Las Vegas, NV, USA, 2020. IEEE. doi: 10.1109/CCWC47524.2020.9031275.
- [17] B. D. Kang, K. H. Jeon, D. Kyoung, et al. Multiple human body tracking using the fusion of ccd and thermal image sensor. In 2011 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), pages 1–4. IEEE, 2011. doi: 10.1109/AIPR.2011.6176356.
- [18] A. Königs and D. Schulz. Evaluation of thermal imaging for people detection in outdoor scenarios. In 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), pages 1–6, College Station, TX, USA, 2012. IEEE. doi: 10.1109/SSRR.2012.6523883.

- [19] C.-B. Liu and N. Ahuja. Vision based fire detection. In Proceedings of the 17th International Conference on Pattern Recognition, volume 4, pages 134–137. IEEE, 2004.
- [20] V. Myrzin, T. Tsoy, Y. Bai, et al. Visual data processing framework for a skin-based human detection. In A. Ronzhin, G. Rigoll, and R. Meshcheryakov, editors, Interactive Collaborative Robotics, pages 138–149. Springer International Publishing, 2021. doi: 10.1007/978-3-030-87725-5 12.
- [21] M. I. Perdana, A. Risnumawan, and I. A. Sulistijono. Automatic aerial victim detection on low-cost thermal camera using convolutional neural network. In 2020 International Symposium on Community-Centric Systems (CcS), pages 1–5, Tokyo, Japan, 2020. IEEE. doi: 10.1109/CcS49175.2020.9231433.
- [22] Rene Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.
- [23] Rene Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 12179–12188, 2021.
- [24] S. K. Sharma, R. Agrawal, S. Srivastava, and et al. Review of human detection techniques in night vision. In 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pages 2216–2220, Chennai, India, 2017. IEEE. doi: 10.1109/WiSPNET.2017.8300153.
- [25] P. Wang, J. Zhang, and H. Zhu. Fire detection in video surveillance using superpixel-based region proposal and ese-shufflenet. Multimedia Tools and Applications, pages 1–28, 2021.
- [26] Y. Xiao, V. R. Kamat, and C. C. Menassa. Human tracking from single rgb-d camera using online learning. Image and Vision Computing, 88:67–75, 2019. doi: 10.1016/j.imavis.2019.03.003.
- [27] Y. Xie, J. Zhu, Y. Guo, J. You, D. Feng, and Y. Cao. Early indoor occluded fire detection based on firelight reflection characteristics. Fire Safety Journal, 128:Article 103542, 2022.
- [28] G. Xu, Y. Zhang, Q. Zhang, G. Lin, Z. Wang, Y. Jia, et al. Video smoke detection based on deep saliency network. Fire Safety Journal, 105:277–285, 2019.

- [29] A. Yazdi, H. Qin, C. B. Jordan, L. Yang, and F. Yan. Nemo: An open-source transformer-supercharged benchmark for fine-grained wildfire smoke detection. *Remote Sensing*, 14(16):3979, 2022.
- [30] T. H. Yoon, K. S. Kim, J. W. Lee, et al. Image segmentation of human forearms in infrared image. In *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2762–2765, New York City, NY, USA, 2006. IEEE. doi: 10.1109/IEMBS.2006.260706.
- [31] F. Yuan, L. Zhang, B. Wan, X. Xia, and J. Shi. Convolutional neural networks based on multi-scale additive merging layers for visual smoke recognition. *Machine Vision and Applications*, 30:345–358, 2019.

10

**ENHANCING INDOOR MAPPING AND LOCALIZATION IN
SPECULAR RICH ENVIRONMENTS USING DEEP LEARNING
AND SENSOR FUSION**

Renatto Tommasi Hernández

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 154

**ENHANCING INDOOR MAPPING AND
LOCALIZATION IN SPECULAR RICH
ENVIRONMENTS USING DEEP LEARNING
AND SENSOR FUSION**

Renatto Tommasi Hernández

Zagreb, June 13, 2025

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 154

**UNAPRJEĐENJE IZRADE KARTE I
LOKALIZACIJE U ZATVORENOM PROSTORU S
IZRAŽENIM ZRCALNIM ODRAZIMA POMOĆU
DUBOKOG UČENJA I FUZIJE SENZORA**

Renatto Tommasi Hernández

Zagreb, lipanj 13, 2025.

First and foremost, I am profoundly grateful to the European Union and its ERASMUS MUNDUS Joint Masters Program for the financial opportunity to join the Intelligent Field Robotic Systems (IFROS) Master's Program. I offer my sincere thanks to the IFROS team in Girona, whose tireless dedication has made this such an exceptional program, and to the LARICS lab at the University of Zagreb for their vital partnership.

I want to offer my special thanks to my supervisors, Prof. Goran Vasiljević (University of Zagreb) and Prof. Zoltán Istenes (ELTE University, Budapest). Their mentorship, insightful perspectives, and consistent support were essential in guiding this thesis from conception to completion. Additionally, I am grateful to the team at EUROKNOWS for the fantastic opportunity to contribute to the Virofighter robot project.

None of this would have been possible without my family. I am eternally grateful for their unwavering support and love, which has been my foundation every step of the way.

Finally, to my girlfriend, whose belief in me never wavered, especially when my own did.

Contents

1	Introduction	3
1.1	Background and Context	3
1.2	Problem Statement	4
1.3	Structure Overview	5
2	Literature Review	7
2.1	Theoretical Framework	7
2.1.1	Specular vs. Diffuse Reflections	7
2.1.2	3D Cameras	8
2.1.3	Wide Field of View Cameras: Camera and Distortion Models	8
2.1.4	Visual Odometry: State Estimation using Cameras	9
2.1.5	Sensor Fusion: Using Pose-Graph Optimization for robust sensor fusion	11
2.1.6	Challenges posed by Specularities	12
2.2	Related Work	13
2.2.1	Dealing with Specularities in the Literature	13
2.2.2	Comparing Different Feature Detectors and Descriptors in Different Environments	15
2.2.3	Monocular Visual Odometry and Wide FoV Advantages	15
3	Methodology	17
3.1	RT-SpecFilter: Segmentation and Filtering of Specular Artifacts	18
3.1.1	Main Flattening Thread	18
3.1.2	Classification and Mapping Thread	20
3.1.3	Training the SVM Classifier	21

3.2	Feature Detector Analysis on Highly Specular Environments	23
3.3	Multicam SP-VO: Leveraging multiple wide FoV cameras for state estimation in highly specular environments	24
3.3.1	Camera Calibration	25
3.3.2	Implementation Details	26
3.3.3	Sensor Fusion using Pose-Graph Optimization	29
4	Results and Analysis	31
4.1	RT-SpecFilter Validation Experiments	31
4.1.1	Gray Floor with Sun Reflections	31
4.1.2	Indoors with strong overhead lights	33
4.2	Feature Detector Comparison for Highly Specular Environments	36
4.3	Multicam SP-VO Validation Experiments	38
4.3.1	Visual Validation for Multicam SP-VO	38
4.3.2	Landmark Drift as Validation for Multicam SP-VO	39
5	Conclusion and Future Work	42
5.1	Limitations and Future Work	43
	References	44
	Abstract	50
	Sažetak	51
	Annex A: Camera Calibration Parameters	52
	Annex B: Landmark Drift Full Results	54

1 Introduction

The world of robotics and physical AI is finally getting a day pass out of the lab into the world. These are exciting times for humans, who are thrilled to finally be able interact with machines that are only known to them through science fiction and imagination. For the machines, not everything is as invigorating; after all, they are used to perfect, controlled environments and the real world is anything but perfect and controlled. With sensors that aren't robust enough to imperfections such as light reflections on the ground, the robot's might not last long outside of the lab.

The following section introduces the efforts undertaken to address the gap in current work aimed at enhancing the robustness of sensors to light reflections on surfaces, specifically on 3D cameras and keypoint detection for visual localization techniques. Section 1.1 will introduce the reasoning behind having to deal with the bad readings created by light sources. Thereafter, Section 1.2 will discuss the particularities of the research problem, while Section 1.3 will elaborate on the organizational structure of the remaining work. This work has the purpose of enhancing an already existing industrial product, dealing with real issues caused by the highly reflective surfaces present in the environments it operates on.

1.1 Background and Context

Quick acceptance of autonomous robots by the public relies mostly on robust systems able to handle real life scenarios as seamlessly as possible. Circular autonomous vacuum cleaners with purely reactive controls come to mind. This design led to the robot charging at obstacles, bumping into them, changing direction, and continuing. Although it eventually finished, there are time sensitive scenarios where that would not be enough. Especially not in quick and dynamic environments where swiftly cleaning and disinfect-

ing a room prove crucial and are directly responsible for how fast the equipment can be used once more. Mobile platforms such as these now use different suites of sensors, paired with robust exploration algorithms that ensure that 100% of the room is covered as efficient and effective as possible.

Time efficient and robust exploration algorithms finally fall to the accuracy of the localization and mapping abilities of the robot. If the map is filled with noise caused by how the sensor interprets the environment around it, the robot might end up avoiding imaginary obstacles. If the localization is off, the robot might never finish exploring the room since its state estimation might lead it to believe to be meters away from the starting point, when in fact, it's crossed it several times by now. Simultaneous Localization and Mapping (SLAM) techniques work on solving the localization of the robot based on the map, which is simultaneously being created thanks to the robot's localization [1]. Many SLAM techniques are then fine-tuned based on the environment, the onboard sensors and system's capabilities.

Still, localization and mapping techniques often struggle in indoor scenarios due to several noise sources, such as moving objects, small obstacles on the ground and light reflections caused by overhead lamps. As a result, there is a need for systems that directly address the noise sources, such as highly reflective surfaces and robustly deal with them [2, 3, 4, 5].

1.2 Problem Statement

This work focuses on the Euroknows Virofighter, shown in Figure 1.1, an autonomous UVC-light disinfection robot equipped with an Intel RealSense D435 RGB-D camera for mapping, wheel encoders for localization and 4 wide field-of-view (FoV) cameras for teleoperation. The virofighter usually works on hospital environments and with the RGB-D Camera facing the ground, it struggles with noise on the depth estimation created from reflective floors and localization drift from the wheel odometry, leading to inefficient or incomplete disinfection. The efforts in this thesis look to tackle said problems, both in localization and mapping by dealing with the following research questions:

1. How can specular reflections' adverse effects on RGB-D camera depth estimation



Figure 1.1: The Euroknows Virofighter, an autonomous UVC-light disinfection robot equipped with an Intel RealSense D435 RGB-D sensor and 4 wide field of view cameras

and visual data quality be mitigated in SLAM systems?

2. How do various feature detectors and descriptors perform in images with specular reflections?
3. How can a multi-camera system with four wide FoV lenses enable accurate visual odometry in specular-rich environments?

To address the research questions, this thesis first introduces the RT-SpecFilter, a real-time post-processing algorithm designed to mitigate the adverse effects of specular reflections on RGB-D camera depth estimation by segmenting and filtering specular artifacts. Subsequently, a comparative analysis of various feature detectors is conducted to evaluate their performance in specular-rich environments. Finally, the thesis proposes a multi-camera visual odometry solution that utilizes four wide FoV lenses and the Super-Point [6] deep learning model to enhance the current state estimation by significantly reducing the orientation drift in these challenging conditions.

1.3 Structure Overview

This work is organized as follows: **Chapter 2** provides a theoretical framework behind the work as well as a literature review of the current state of the art in dealing with noise

caused by light sources in field robotics. This will be mostly focused on 3D sensors and wide field of view cameras, since those will be the sensors used during the rest of the work. **Chapter 3** will go into the detail of the proposed approach when dealing with light reflections. A filter for the D435 Intel Realsense 3D camera will be introduced, as well as an analysis on the robustness of common feature detectors against highly specular surfaces. Finally, a Multi-Camera Visual Odometry system, named Multicam SP-VO, will use the insights of the previous study to create a more robust state estimation technique. **Chapter 4** will showcase the results obtained by several experiments to which the complete system was subjected. Finally, **Chapter 5** will wrap up the implemented proposal and suggest future work to further address the challenges into creating an even more robust localization and mapping system.

2 Literature Review

The following chapter will focus on establishing the foundation of this work by first going into the foundational theory within section 2.1: Theoretical Framework. The rest of the work will be based on such principles. Section 2.2, the current state of the art corresponding to this work's context will be introduced.

2.1 Theoretical Framework

The theoretical framework builds on basic knowledge of computer vision, camera modeling, state estimation and sensor fusion, going deeper into the most relevant areas for this thesis such as Specular vs. Diffuse Reflections (section 2.1.1), 3D Cameras (section 2.1.2), wide FoV cameras (section 2.1.3), Visual Odometry (section 2.1.4), Pose Graphs (section 2.1.5) and the challenges of specular reflections (section 2.1.6).

2.1.1 Specular vs. Diffuse Reflections

Specular reflections occur when light from a source reflects off an object's surface in a mirror-like manner, preserving the color and intensity of the light source. In contrast, diffuse reflections scatter light in multiple directions, revealing the object's intrinsic "body color" influenced by both the light source and the surface material. The Dichromatic Reflection Model (DRM), proposed by Shafer et al. [7], describes surface reflection as a linear combination of these specular and diffuse components for most inhomogeneous materials. This model provides a theoretical foundation for separating and analyzing reflection components, which is critical for mitigating specular effects in imaging systems. In this context, a glossy object would then reflect more of the specular component, while a matte finished surface would reflect a higher diffuse component in comparison. This work will refer to reflections with a high specular value as *specularities*, but it is

important to note that all reflections contain both components.

2.1.2 3D Cameras

RGB-D cameras combine conventional color imaging with depth sensing capabilities, enabling three-dimensional scene capture. These cameras have become essential in robotics, computer vision, and augmented reality applications. The depth component is typically acquired through one of two primary technologies:

- *Structured-light*: These systems project known light patterns onto the scene and calculate depth based on the deformation of these patterns when they fall on objects at varying distances[8].
- *Time-of-flight (ToF)*: These measure the time taken for emitted light to bounce off objects and return to the sensor, thereby calculating distance[8].

2.1.3 Wide Field of View Cameras: Camera and Distortion Models

Given that the proposed visual odometry system in this thesis relies on multiple wide FoV cameras (as detailed in Section 3.3.2), an appropriate camera model is essential for accurate feature localization and motion estimation. Standard pinhole camera models, while suitable for narrow-FoV lenses due to their assumption of linear projection, are insufficient for wide FoV lenses which exhibit significant radial distortion [9]. To address this, this thesis employs the pinhole-equidistant camera model, a common and effective approach for such lenses.

Pinhole-Equidistant Camera Model

The pinhole-equidistant model, notably discussed by Scaramuzza [9], was selected for calibrating the wide FoV cameras (Section 3.3.1). This model provides a good balance between accurately representing camera distortion and maintaining computational simplicity, which is beneficial for real-time visual odometry pipelines [9]. Its key principles, relevant to how feature points are processed in our system (Section 3.3.2), are:

1. **Pinhole Projection**: Initially, a 3D point is projected onto the image plane us-

ing the ideal pinhole camera model, defined by intrinsic parameters: focal lengths (f_x, f_y) and the principal point (c_x, c_y) [9].

2. **Equidistant Distortion Correction:** Radial distortion, prevalent in wide FoV lenses, is then corrected. The equidistant model maps the angle of incidence θ (between the camera's optical axis and the incoming light ray from the 3D point) to a radial distance r in the image plane according to the linear relationship $r = f\theta$, where f is a wide FoV-specific focal length parameter [9]. This property aims to preserve angular distances. Deviations from the ideal equidistant projection are accounted for by modeling the distorted radial distance r_d using a polynomial function of θ :

$$r_d = a_0 + a_1\theta + a_2\theta^2 + a_3\theta^3 + \dots \quad (2.1)$$

where a_i are the distortion coefficients estimated during the calibration process [9].

This model effectively handles the strong non-linear projections characteristic of wide FoV lenses, where straight lines in the 3D world often appear curved in the image [9]. Its computational efficiency compared to some other omnidirectional models was a factor in its adoption for the visual odometry pipeline developed in this thesis [9].

2.1.4 Visual Odometry: State Estimation using Cameras

Visual Odometry (VO) is a process that estimates a camera's ego-motion by analyzing sequential images. This technique is foundational in robotics, enabling navigation using camera data alone or in conjunction with other sensors [10, 11]. VO frameworks typically operate by extracting and matching salient features between consecutive image frames, f_{k-1} and f_k , to determine the relative camera transformation T_k [12]. Depending on whether these feature correspondences are represented in 2D image coordinates or 3D world coordinates, VO can be broadly categorized into three main motion estimation strategies: 2D-2D, 3D-2D, and 3D-3D [12]. Given that this thesis ultimately employs a multi-camera 2D-2D motion estimation strategy, this is of primary relevance.

2D-2D Motion Estimation

In the 2D-2D case, feature correspondences between frames f_{k-1} and f_k are specified entirely in 2D image coordinates. The relative camera motion, consisting of rotation $R_{k,k-1}$

and translation $t_{k,k-1}$, is determined through the Essential Matrix, E . The Essential Matrix encapsulates the epipolar geometry between two calibrated camera views, relating corresponding normalized image points $p_k = [u_k, v_k, 1]^T$ and $p'_k = [u'_k, v'_k, 1]^T$ via the epipolar constraint:

$$p_k'^T E p_k = 0 \quad (2.2)$$

Expanding this equation, where $E = [e_{ij}]$, we get:

$$u'_k u_k e_{11} + u'_k v_k e_{12} + u'_k e_{13} + v'_k u_k e_{21} + v'_k v_k e_{22} + v'_k e_{23} + u_k e_{31} + v_k e_{32} + e_{33} = 0 \quad (2.3)$$

This equation is linear in the nine unknown elements of E , requiring a minimum of 5 points [13]. Once E is computed from a set of 2D correspondences, the rotation $R_{k,k-1}$ and the *direction* of the translation vector $t_{k,k-1}$ can be extracted.

A critical characteristic of monocular 2D-2D VO is that the absolute scale of the translation $t_{k,k-1}$ is unobservable from two views alone; only its direction can be determined. This scale ambiguity means the Essential Matrix has 5 Degrees-of-Freedom (DoF), even though full 3D motion has 6 DoF. To recover the full metric pose, the scale must be inferred, often by integrating data from other sensors (such as wheel encoders, as used in this thesis) or by tracking points over multiple views. Robust estimation of E in the presence of outlier feature matches is typically achieved using Random Sample Consensus (RANSAC) [12, 14, 15].

Alternative Motion Estimation Approaches

- **3D-2D Motion Estimation (Perspective-n-Point - PnP):** This method estimates pose when a set of 3D points X_{k-1} (e.g., from stereo triangulation or a depth sensor) and their corresponding 2D projections p_k in the current image are known [12]. The camera pose T_k is found by minimizing the reprojection error of these 3D points onto the image plane. While potentially offering direct scale information if the 3D points are metric, PnP requires reliable 3D point data, which can be challenging to obtain consistently, especially in specular environments or with limited stereo overlap [12]. The general form of the PnP problem is to find T_k that

minimizes:

$$\sum_i ||p_k^i - \hat{p}(X_{k-1}^i, T_k)||^2 \quad (2.4)$$

where $\hat{p}(X_{k-1}^i, T_k)$ is the projection of X_{k-1}^i using T_k .

- **3D-3D Motion Estimation (Point Cloud Registration):** This approach aligns two sets of corresponding 3D points, $\tilde{X}_{k-1}^i$ and $\tilde{X}_k^i$, to find the transformation T_k [12]. It requires 3D coordinates for features in both viewpoints, typically obtained from stereo systems or depth sensors, with a minimum of three non-collinear correspondences needed [12]. The objective is to minimize:

$$\sum_i ||\tilde{X}_k^i - T_k \tilde{X}_{k-1}^i||^2 \quad (2.5)$$

Back-end Optimization

Regardless of the front-end motion estimation method, the resulting sequence of pose estimates often accumulates drift over time. Modern VO systems employ back-end optimization techniques to refine these estimates and improve global consistency [15]. These include:

- **Local Optimization (Windowed Pose-Graph Optimization):** Optimizes a recent window of poses, offering computational efficiency but potentially allowing global drift to accumulate [15].
- **Global Optimization (Full Pose-Graph Smoothing):** Optimizes the entire trajectory, capable of significantly correcting drift, especially when loop closures are detected [15].

2.1.5 Sensor Fusion: Using Pose-Graph Optimization for robust sensor fusion

Pose-graph optimization offers a powerful and flexible framework for robustly fusing data from multiple sensors to achieve accurate and consistent state estimation [16, 17, 18]. In this context, the system's pose is represented as nodes in a graph. Measurements from various sensors, which provide information about the relative motion be-

tween these poses or absolute constraints on a pose, are incorporated as factors (edges) connecting these nodes. Each factor encodes a probabilistic constraint, reflecting the measurement and its associated uncertainty.

An efficient approach to solving such sensor fusion problems formulated as pose graphs is iSAM2 (Incremental Smoothing and Mapping) [19]. While often associated with SLAM, its core principles are highly applicable to sensor fusion focused on pose estimation. iSAM2 incrementally builds and updates a Bayes Tree, which represents the probabilistic relationships between all estimated poses as defined by the factors derived from the various fused sensor measurements [19]. iSAM2 efficiently updates only the affected parts of the Bayes Tree and performs "fluid relinearization" of pose variables whose estimates change significantly. This allows it to maintain a globally consistent estimate of the entire pose trajectory, fusing all sensor information over time, with significantly reduced computational overhead compared to traditional Global Optimization [19]. It finds the set of poses that best satisfies all the multi-modal sensor constraints according to their specified uncertainties, thereby providing a robust and coherent fused state estimate.

2.1.6 Challenges posed by Specularities

Specularities can pose different problems depending on the system used, this section will cover only challenges for 3D cameras and feature detection.

Distortion of Depth Information in 3D Sensors

Specular reflections pose a significant challenge to the performance of 3D cameras. When light encounters a highly reflective surface—such as polished metal, glass, or wet tissue—it undergoes specular reflection, redirecting the beam in a single, mirror-like direction rather than scattering diffusely back to the sensor. This redirection can prevent the sensor from receiving the expected return patterns, introducing noise or erroneous depth values [3, 4].

The Intel RealSense camera used in this thesis, a popular structured-light RGB-D sensor for robotics applications, suffers from these challenges. Its depth maps frequently exhibit artifacts in regions of high specularity, such as gaps where no depth is recorded or

spurious values that deviate from the true surface geometry [20]. These distortions compromise the accuracy of 3D reconstructions. Quantitatively, studies have shown that specular reflections can significantly increase depth error rates in affected regions (depending on surface reflectivity and sensor configuration) [4], underscoring the need for robust mitigation strategies in RGB-D-based systems.

Challenges for feature detection and tracking

Specular surfaces present formidable challenges in computer vision, particularly for tasks such as keypoint detection and description, due to their non-Lambertian, reflective properties. Unlike Lambertian surfaces, which maintain a uniform appearance regardless of viewpoint, specular surfaces produce concentrated light reflections, resulting in view-dependent highlights [7]. These highlights introduce variability in the position of such highly distinguishable surfaces, which is foundational to many feature detection and matching algorithms. Such assumptions are vital for applications like visual tracking, feature matching, and 3D reconstruction, which depend on identifying stable, repeatable keypoints across multiple images.

2.2 Related Work

The following section showcases recent works in dealing with specularities, feature detectors and monocular visual odometry.

2.2.1 Dealing with Specularities in the Literature

Several studies in computer vision have examined specular removal, which is crucial for object detection and recognition. The following section compiles the most relevant ones for the present work, dividing the challenge into 2 problems: identifying the reflection and dealing with it.

Identifying Specular Reflections

Detecting specular reflections is a critical step in mitigating their effects. Traditional methods apply thresholding in color spaces such as RGB or HSV. In RGB, specular highlights are identified as pixels with near-saturated values across all channels (e.g., [255,

255, 255]). In HSV, specularities correspond to pixels with high brightness (value) and low saturation, offering two criteria for detection [21]. Post-thresholding, image gradient analysis around bright regions helps distinguish specular highlights from uniformly bright surfaces, reducing false positives [21].

Advanced approaches employ clustering techniques, such as Gaussian Mixture Models [5] or k-means clustering [22], to classify pixels into specular and diffuse categories based on intensity or color features. These methods group specular pixels into high-intensity clusters, improving detection accuracy. More recently, deep learning techniques have gained traction, leveraging complex feature learning to robustly segment specular regions in diverse real-world scenarios [5, 23]. These methods outperform classical approaches by adapting to varying lighting conditions and surface properties.

3D Sensor Handling of Specularities

For RGB-D sensors, Or-El et al. [24] address specular reflections in structured light systems like Intel RealSense cameras. They propose a real-time depth refinement model that incorporates specular reflections and multiple albedos, using the camera's monochromatic IR projector and IR images to reconstruct accurate depth maps. Yoshida et al. [25] tackle ToF sensor limitations, where specular surfaces cause depth data holes. Their method propagates sparse ToF depth values within specular regions using polarization cues and a superpixel-based approach, assuming planar surface approximations to resolve azimuth ambiguities.

Beyond RGB-D cameras, Koch et al. [26] focus on multi-echo laser scanners, which face similar challenges with reflective surfaces like glass and mirrors. They introduce a two-stage filtering approach: an on-the-fly filter to reduce errors in individual scans and a robust filter that refines registered maps using multiple scans. While not directly applied to RGB-D cameras, their techniques inspire multi-stage strategies for specular mitigation.

2.2.2 Comparing Different Feature Detectors and Descriptors in Different Environments

The selection of an appropriate feature detector and descriptor is crucial for robust visual odometry, particularly in challenging specular environments. Table 2.1 summarizes key characteristics of SIFT, ORB, AKAZE, and SuperPoint, with an emphasis on aspects relevant to the Virofighter’s operational context and the methodology adopted in this thesis (Sections 3.2 and 3.3.2).

2.2.3 Monocular Visual Odometry and Wide FoV Advantages

Monocular Visual Odometry (VO) has seen significant advancements. Semi-direct Visual Odometry (SVO) [32] combines direct tracking with feature-based optimization, enabling robust performance even in texture-poor areas, and has been extended to Visual-Inertial SLAM for scale estimation. Deep learning approaches have also gained prominence. Tateno et al. [33] utilized Convolutional Neural Networks (CNNs) for depth estimation within a monocular SLAM framework, demonstrating improved robustness in challenging lighting and feature-scarce conditions. Further research from TUM, including MonoRec [34] for semi-supervised dense reconstruction and D3VO [35] for deep learning-based depth, pose, and uncertainty estimation, underscores the potential of learned methods. LSD-SLAM [36] introduced direct tracking with semi-dense depth mapping and explicit scale drift correction in pose-graph optimization. While these methods showcase advancements in handling complex scenarios, their specific application to environments dominated by specular reflections, a key challenge in this thesis, often remains an open question, suggesting an area where further investigation into deep learning’s role could be beneficial.

Advantage of Wide FoV Cameras

The utility of wide FoV cameras for VO, particularly in constrained indoor environments relevant to this thesis, is supported by research such as Zhang et al. [37]. Their findings indicate that large FoV cameras can offer superior performance indoors compared to narrower FoV alternatives.

Feature	Key Principle / Type	Robustness Highlights	Computational Cost / Speed	Relevance to Specular Environments & Methodology
SIFT [27]	Scale-space extrema (DoG) detector; Gradient orientation histograms descriptor (128-float).	Invariant to scale, rotation; Robust to affine distortion, noise, and illumination changes [28].	Computationally more expensive than ORB or AKAZE [29, 30].	Robustness to illumination is valuable, but high computational cost makes it less ideal for real-time robotics. Serves as a performance benchmark.
ORB [29]	oFAST (oriented FAST) corner detector; Rotated BRIEF binary descriptor.	Rotation invariant; Relatively immune to Gaussian noise [29]. FAST-based detectors can be less reliable under large image transformations compared to SIFT.	Fast, designed for real-time applications; Significantly faster than SIFT [28].	Speed is advantageous for robotics. Immunity to Gaussian noise is good, but its performance in highly specular conditions with dynamic illumination changes is less emphasized compared to others.
AKAZE [31]	Detector uses maxima of Hessian response in nonlinear scale space (non-linear diffusion); M-LDB binary descriptor.	Preserves boundaries while reducing noise; Generally better repeatability under noise, blurring, and illumination variations compared to Gaussian scale-space methods [28, 30]. Scale and rotation invariant.	Faster than SIFT/SURF, but typically less computationally efficient than ORB [29].	Improved repeatability under illumination variations is highly relevant for specular environments. Balances robustness and efficiency.
SuperPoint [6]	Fully-convolutional neural network; Joint detection and description; Self-supervised training with Homographic Adaptation.	Improved repeatability, especially under challenging conditions like illumination changes (explicitly beneficial in specular environments) [6]. Provides dense and accurate matches. Performs comparably to or better than SIFT/ORB in geometric matching tasks [6].	Capable of real-time performance [6].	Its strong performance in challenging illumination (key for specularity) and real-time capability make it highly suitable for the thesis goals.

Table 2.1: Comparison of Key Feature Detectors and Descriptors.

3 Methodology

The purpose of this thesis is to create a system that is robust against specularities when using camera based sensors such as the Intel Realsense D435 camera and wide FoV cameras in indoor environments with strong overhead illumination. Building on previous work and theoretical foundation, the work will be separated into the following 3 sections: Section 3.1 deals directly with 3D artifacts created by specularities detected by the RGB-D Camera by proposing a Real Time Specularity Filter (RT-SpecFilter), Section 3.2 that fills the gap of finding a feature detector that is robust against detecting keypoints in specularities, and finally, Section 3.3 that uses the previously identified feature detector to design a visual odometry pipeline (Multicam SP-VO) that leverages a 4 wide FoV camera system. The diagram of the final system can be observed in Figure 3.1.

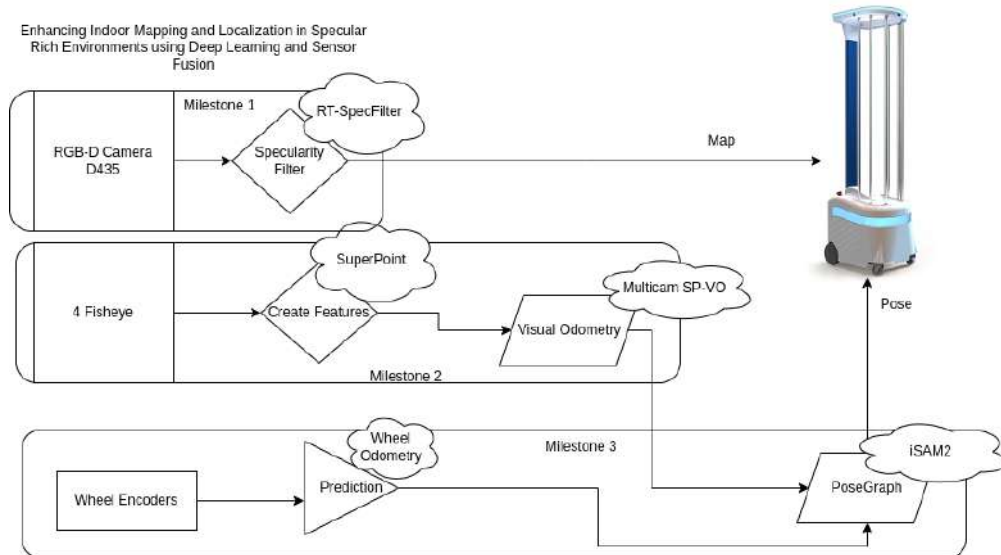


Figure 3.1: System Diagram

3.1 RT-SpecFilter: Segmentation and Filtering of Specular Artifacts

This section details the RT-SpecFilter, a real-time post-processing algorithm proposed to mitigate specular artifacts in RGB-D point clouds from the Intel RealSense D435 sensor. The approach leverages extrinsic camera-robot calibration and is inspired by prior work by Yoshida et al. [25] and Koch et al.[26]. RT-SpecFilter operates using two concurrent threads: a main flattening thread for immediate artifact handling and a classification and mapping thread for robust, feature-based analysis.

This dual-threaded design, integrating rapid geometric segmentation and flattening with robust, feature-based classification and dynamic mapping, allows RT-SpecFilter to effectively mitigate specular artifacts. This, in turn, enhances the accuracy and reliability of the Virofighter’s SLAM capabilities in challenging indoor environments prone to reflections. The overall workflow is depicted in Figure 3.2.

3.1.1 Main Flattening Thread

The main thread focuses on low-latency processing of incoming RGB-D point clouds to identify and handle potential specular artifacts in real-time.

Artifact Segmentation

Potential specular artifacts are initially segmented based on their 3D voxel positions relative to the robot’s base frame (as illustrated in Figure 3.3, where the specular artifact can be seen under the xy plane defined by the blue grid). Using the known extrinsic calibra-

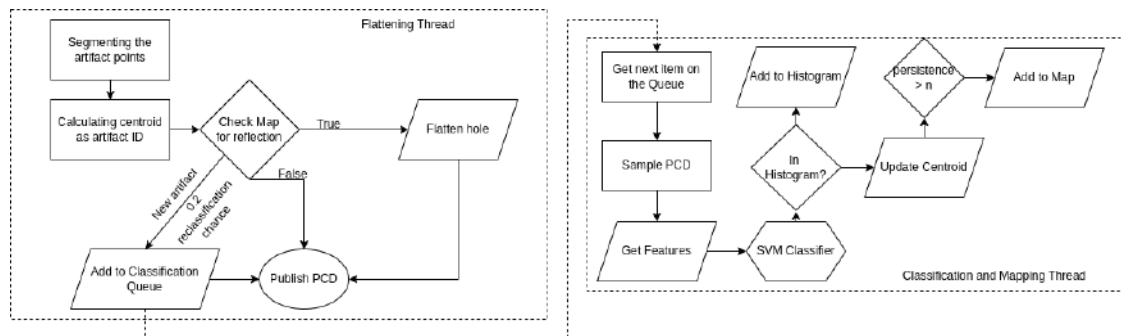


Figure 3.2: Diagram of the RT-SpecFilter algorithm, illustrating the interaction between the main flattening thread and the classification and mapping thread.

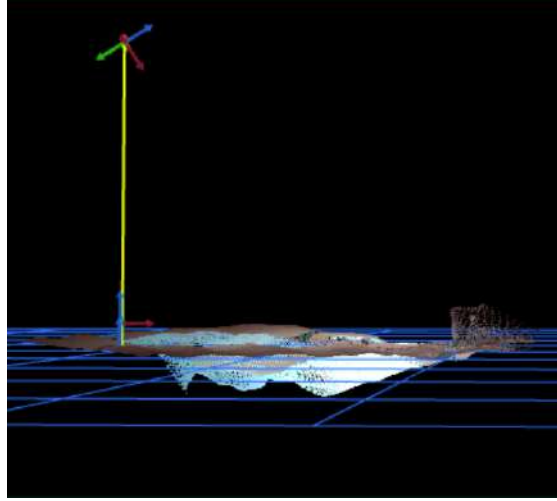


Figure 3.3: Point cloud from the Intel RealSense D435, showing a specular artifact relative to the robot's base frame. The camera, mounted atop the robot, points downward, with the top frame illustrating the camera frame and the bottom frame representing the base frame.

tion, points falling below a predefined ground-plane threshold (z_{th}), typically set at -0.05 meters in the robot's base frame, are considered part of an artifact. This targets specular-induced "holes" while accounting for minor noise. An example of this segmentation is shown in Figure 3.4.

Artifact Flattening and Immediate Classification

For each segmented artifact, its centroid is computed to serve as a unique identifier. This centroid is queried against a pre-existing map of known artifacts (stored as centroid-classification pairs). If a matching centroid (within a 0.5-meter Euclidean distance) is found and its stored label is "specular," the artifact's Z-coordinates (in the robot frame)

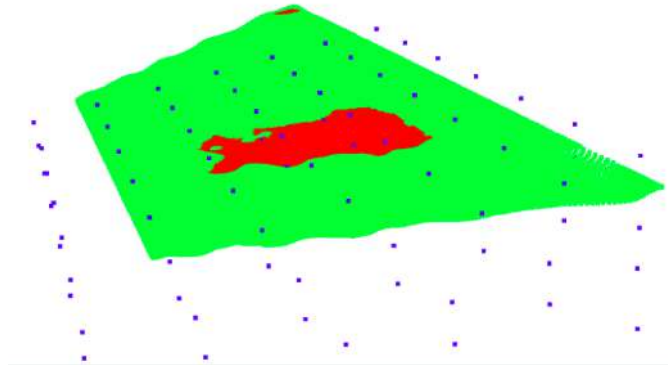


Figure 3.4: Segmentation algorithm identifying possible specular artifact (in red) based on a $z_{th} \leq -0.05$ threshold on the base frame

are adjusted to z_{th} . This "flattening" corrects the erroneous depth, allowing the robot to treat the area as navigable.

If no match is found, the artifact is enqueued in a classification queue for processing by the secondary thread. Non-specular artifacts, as determined by the map or subsequent classification, are passed to the mapping pipeline without modification. The flattening process is optimized for low latency (typically 15 Hz), ensuring seamless integration with the Virofighter's autonomous navigation.

3.1.2 Classification and Mapping Thread

Operating at a lower frequency (e.g., 1 Hz) to offload computationally intensive tasks, this thread performs detailed analysis of enqueued artifacts to determine their nature and update a persistent map.

Feature Extraction

Enqueued artifacts are characterized by extracting a 9-dimensional feature vector. This includes:

- **Mean HSV Values:** RGB points are converted to HSV, and mean Hue, Saturation, and Value are computed. Low saturation and high value are indicative of specular reflections, consistent with prior work by Morgand and Tamaazousti [21].
- **Surface Normal Statistics:** To leverage 3D data, surface normals are estimated using PCA (search radius 0.1m), and their mean vector (n_x, n_y, n_z) and standard deviation are calculated. Uneven normal distributions can indicate specular artifacts compared to smoother, actual geometry.
- **Mean Normal Orientation:** The azimuth (θ) and elevation (ϕ) angles of the mean normal are also included.

This feature vector is represented as a 1×9 matrix of 32-bit floats.

SVM-Based Classification

The feature vector is processed by a pre-trained Support Vector Machine (SVM) [38], which creates a hyperplane for classification, to classify between regions to be flattened or not. Details on the training of the SVM are in Section 3.1.3. The SVM was selected for its fast computation and classification clarity. An RBF kernel addresses data nonlinearity, and normalization prevents feature bias[38].

Dynamic Histogram and Map Updating

To adapt to the dynamic nature of specular reflections (shifting with robot/light source motion) and possible misclassifications, a histogram-based mapping system is employed. The centroid and SVM classification of each processed artifact are added to a histogram that tracks observation frequency. Centroids are updated as the mean of current and historical observations to mitigate noise and positional shifts. After an artifact is consistently observed and classified $n=5$ times (a tunable parameter for reliability), its centroid and classification are committed to the permanent map. This map enables efficient lookups by the Main Flattening Thread.

3.1.3 Training the SVM Classifier

Prior to discussing the training setup in detail, it is imperative to define our state space. In the context of the RGB-D Specular configuration on the Virofighter, it is characterized as follows:

$$\text{state} = \{r \quad h \quad | \quad c\} \quad (3.1)$$

The state is delineated by the environmental conditions, utilizing two binary independent variables to denote the presence of specular reflections in the image (r) and the existence of a hole on the ground (h). The features are subsequently computed as described in Section 3.1.2, input into the SVM, which then provides a classification (c). The resultant state space, composed of two binary variables, has a size of $2 + 2 = 4$. Each state is accompanied by a prescribed output, aligned with the desired behavior of the mobile platform. The state space and corresponding responses are encapsulated in the

following truth table:

State		Output
r (reflection)	h (hole)	c (flatten)
r	h	$\bar{c}$
r	$\bar{h}$	c
$\bar{r}$	h	$\bar{c}$
$\bar{r}$	$\bar{h}$	$\bar{c}$

(3.2)

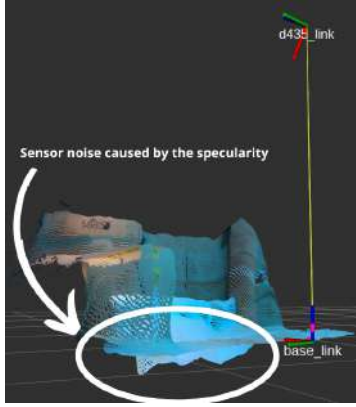
The SVM output does not inherently indicate the presence of a specular reflection but rather determines whether the segmented hole should be flattened. Given that our current implementation segments the 'artifact' based on the Z-value relative to the robot's base frame, it is conceivable for both a reflection and a hole to coexist in the same segmented point cloud. This is further considered in the future work in Chapter 5.

The training environment will be on a test bench created with all 4 possible states (Figure 3.5). During training, the robot is rotated around its axis, applying the RT-SpecFilter to each state. The features pertaining to each scenario are captured and labeled with the wished label. The SVM is then trained on these features and labels.

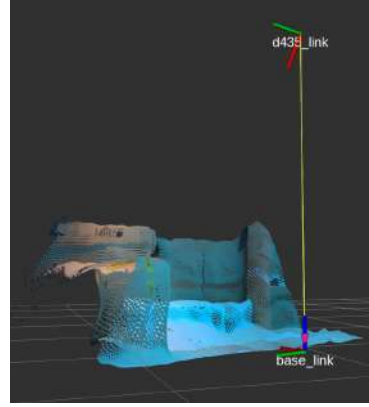
The RT-SpecFilter algorithm was trained under diverse lighting conditions—no light, low light, well-lit, and the UVC lamps where also turned on and off to ensure its performance in detecting and mitigating specular artifacts. The Intel RealSense D435 RGB-D camera was oriented toward the four edges of the test bench at various angles to ensure comprehensive data collection. Figure 3.6 shows an example of a specular reflection with no physical hole before and after filter application, while Figure 3.6c shows an instance with both a specular reflection and a physical hole.



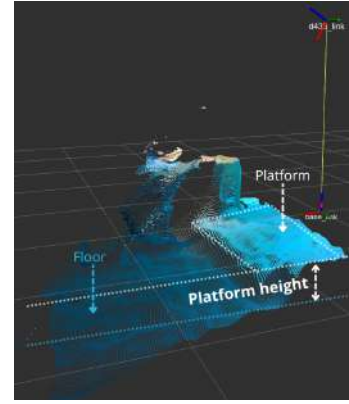
Figure 3.5: Test bench used to train and validate the SVM in the RT-SpecFilter



(a) Specular reflection before filtering.



(b) Specular reflection after filtering.



(c) How the system senses the unevenness between the platform and the ground.

Figure 3.6: Comparison of a specular reflection on the test bench edge before and after applying RT-SpecFilter (a and b). The unevenness of the platform against the floor is shown for reference (c). The images were captured in low light conditions with UVC lamps on the test bench shown in Fig. 3.5.

3.2 Feature Detector Analysis on Highly Specular Environments

The literature highlights the robustness of SIFT, ORB, AKAZE, and SuperPoint to various image transformations, but their performance in specular-rich environments remains inadequately studied due to dataset limitations [2]. This thesis addresses this gap by conducting a comparative analysis of these algorithms on the 3D-Ref dataset [2], which contains images with highly specular reflections.

Based on the work of Morgand and Tamaazousti [21], an RGB image will be segmented into highly specular and low specular regions by means of a binary map. Each

of the researched feature detectors will then be run on such images. The target of this study is to analyze how many of the keypoints were found within a highly specular reflection. The results will then be analyzed and the most robust feature detector will be used in the following section, as a means of creating a more robust visual odometry pipeline.

The detection process is multi-staged [21]:

1. **Contrast Equalization:** Image brightness is adjusted to manage variations in lighting and reduce oversaturation.
2. **Adaptive Thresholding:** Specularities are isolated using a dynamic Value threshold combined with a constant Saturation threshold, designed to adapt to different illumination conditions more effectively than static methods.
3. **Gradient-Based Post-processing:** Candidate specular regions are further refined by analyzing their area evolution under varying thresholds, helping to discard false positives.



Figure 3.7: Specular mask created on parking lot image taken from [39]. Detected Highly Specular areas in red.

3.3 Multicam SP-VO: Leveraging multiple wide FoV cameras for state estimation in highly specular environments

This section proposes a solution to Multi-Camera visual odometry in highly specular environments, named *Multicam SP-VO*. The solution will be an implementation of a feature-based VO pipeline using the SuperPoint Deep Learning Model [6] as means of feature extraction and description. This decision is based on Sections 3.2 and 4.2 of this thesis, proving its efficiency and robustness in real time ($>30\text{Hz}$) and highly specular en-

vironments. This section goes from the calibration of the cameras, to the several motion estimation attempts for this system.

3.3.1 Camera Calibration

Accurate measurements in a multi-camera system require calibration of both intrinsic and extrinsic camera parameters. Intrinsic calibration corrects lens distortions and camera-specific properties, while extrinsic calibration defines the spatial relationships between cameras and the robot's base frame. This enables measurements from each camera to be transformed into the robot's coordinate system using efficient transformations.

Multi-Camera Calibration with Kalibr

The Kalibr [40] toolbox facilitated this process for the system's four wide FoV cameras and the single RGB-D camera. The wide FoV cameras were modeled using a pinhole model with an equidistant distortion profile to account for significant lens distortions, as illustrated by the correction of distortion in an image taken from such sensor in Figure 3.8. After undistorting the image using the distortion parameters, we can observe that straight lines, such as those in the ground of the image, go back to being straight in the image as well.

Extrinsic calibration with Kalibr determined the relative poses between all cameras. These transformations, stored in a YAML file, enable the projection of measurements into a common reference frame via TF2 ROS messages. The RGB-D camera served as the reference frame due to its alignment with the robot's base Z-axis and the utility of its depth data for robust calibration. Detailed calibration parameters are available in Annex A.

The extrinsic relationship between the RGB-D camera and the robot's base frame was established by estimating the ground plane using RANSAC on the camera's point cloud. The plane's normal vector N_C informed the camera's pitch angle, while roll and yaw were assumed to be zero. The camera's height d_C provided the Z-translation T_Z , with X and Y

translations assumed negligible. This yielded the transformation ${}^B T_C$:

$${}^B T_C = \begin{bmatrix} R & T \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad R = R_Z(0) \cdot R_Y(\text{pitch}) \cdot R_X(0), \quad T = \begin{bmatrix} 0 \\ 0 \\ T_Z \end{bmatrix} \quad (3.3)$$



(a) Distorted image before calibration



(b) Undistorted image after intrinsic calibration

Figure 3.8: Comparison of a distorted image and its undistorted counterpart after intrinsic calibration, highlighting the correction of feature points.

3.3.2 Implementation Details

The Virofighter is a non-holonomic mobile platform, constrained to a 3-DoF motion in the 2D plane. Its pose can be defined by the state vector $\mathbf{p} = [x \ y \ \theta]^T$, representing its Cartesian coordinates and orientation. The associated state uncertainty can be modeled by a 3×3 covariance matrix F :

$$F = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_y^2 & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_\theta^2 \end{bmatrix} \quad (3.4)$$

where diagonal terms represent variances and off-diagonal terms represent covariances.

The visual odometry pipeline starts with loading the previously determined camera calibration parameters. Subsequently, for each incoming pair of frames, f_{k-1} and f_k , keypoints are extracted. For this task, the SuperPoint deep learning model [6] is utilized, which was identified in Section 3.2 for its robustness, particularly on surfaces exhibit-

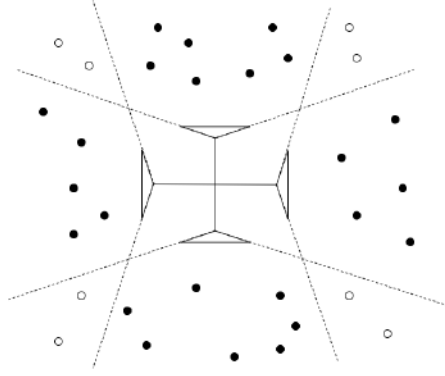


Figure 3.9: Diagram of the calibrated multi-camera system mounted on the Virofighter mobile base. The illustration highlights partial FoV overlap: features depicted in black are visible to a single camera, whereas features in white are simultaneously observable by two cameras.

ing specular reflections. Once keypoints and their descriptors are obtained, a FLANN-based (Fast Library for Approximate Nearest Neighbors) feature matcher is employed to establish correspondences between the keypoints in frame f_{k-1} and f_k . With these correspondences established, the subsequent step is motion estimation.

Two primary methodologies were explored for developing the Multicam SP-VO pipeline. The first centered on a **3D-2D motion estimation** strategy, aiming to capitalize on the overlapping fields of view between cameras. The second, a **2D-2D motion estimation** approach, was designed to utilize wheel odometry data to resolve the inherent scale ambiguity in translation. A simplified schematic of the calibrated multi-camera system is presented in Figure 3.9.

Multi-Camera 3D-2D Motion Estimation Approach

An initial strategy explored for multi-camera visual odometry was a 3D-2D motion estimation approach. This involved attempting to reconstruct 3D points by triangulating features from overlapping regions of adjacent wide FoV camera pairs. The concept was to then fuse independent motion estimates from these pairs.

However, when solving the PnP problem using RANSAC, the model frequently failed to secure enough non-collinear 3D-2D correspondences for robust motion tracking. A modified three-camera stereo configuration (Figure 3.10) was also devised to improve 3D point generation for the PnP algorithm. Despite mitigating the issue of insufficient points to some extent, this PnP-based method still struggled with convergence, often leading to

inaccurate or erratic motion estimates and high latency.

Consequently, this 3D-2D PnP-based approach was deemed unsuitable for the Virofighter's real-time operational requirements in its current form. Insights from literature, such as the use of Bundle Adjustment in systems like Seok and Lim [41], suggest avenues for future refinement if a 3D-2D approach were to be revisited.

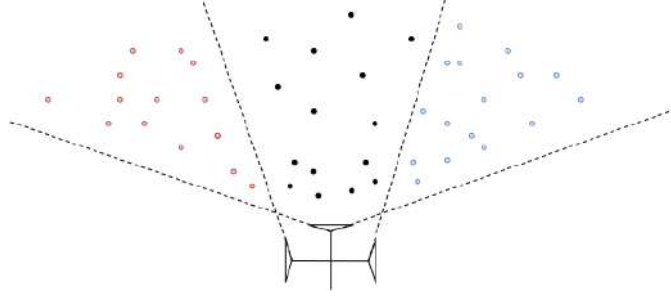


Figure 3.10: Visualization of a three-camera stereo system concept. Features shown in red are hypothetically triangulated by the left stereo pair (e.g., camera 1 and camera 2), while blue features are triangulated by the right stereo pair (e.g., camera 2 and camera 3). Black features represent points visible only within the central camera's frame (camera 2). The blue and red 3D points would then theoretically be projected and tracked on the central camera frame

Multi-Camera 2D-2D Motion Estimation Approach

Initially conceptualized as a proof-of-concept, the 2D-2D motion estimation method ultimately proved to be the most reliable and computationally efficient strategy. Its primary trade-off is the necessity to recover the metric scale of translation from wheel odometry, a source known for its propensity to drift over time. As detailed in Section 2.1, this approach focuses on recovering the Essential Matrix (E) between consecutive frames. Using the previously obtained camera calibration parameters, the points can be normalized and the relative motion can be extracted from E .

Performing this procedure for each of the four cameras yields multiple independent estimates of the robot's motion. Theoretically, these estimates should be largely consistent. This thesis proposes a method to intelligently fuse these estimates by weighting each motion based on its quality. Specifically, the mean reprojection error of the inlier feature correspondences, tracked from frame f_{k-1} to f_k , is used as an inverse measure of confidence for each camera's motion estimate during the subsequent sensor fusion step.

3.3.3 Sensor Fusion using Pose-Graph Optimization

The sensor fusion framework is implemented using pose-graph optimization. The pose-graph is incrementally constructed by adding factors that represent constraints between robot poses. Each valid motion estimate, whether from wheel odometry or visual odometry, contributes a factor to this graph, as illustrated conceptually in Figure 3.11.

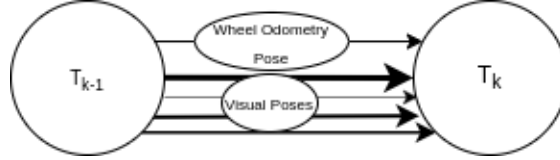


Figure 3.11: Conceptual representation of a pose-graph, illustrating a transformation (factor) between pose T_{k-1} and pose T_k . The thickness in the arrows is proportional to the uncertainty of the pose.

Wheel odometry measurements are integrated between keyframes, which are typically established by the visual odometry pipeline. The factor derived from wheel odometry incorporates a constant uncertainty, modeled by the covariance matrix F_W :

$$F_W = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.10 \end{bmatrix} \quad (3.5)$$

This matrix structure is a deliberate modeling choice reflecting the inherent error characteristics of wheel odometry, where the variance for orientation (yaw) is set to 0.10, an order of magnitude larger than the 0.01 variance for the translational (x and y) components. This decision accounts for the fact that wheel odometry is known for its propensity to drift, as small physical errors from sources like wheel slippage or surface irregularities accumulate most significantly into the robot's heading. This makes orientation the least certain component of the estimate, a fact that is used to correctly weight it against more precise measurements from the visual odometry system during sensor fusion. The off-diagonal zeros further represent a simplifying assumption that the errors in translation and rotation are uncorrelated for a single measurement step.

In contrast, factors derived from visual odometry (one for each of the active cameras) possess variable uncertainty. This uncertainty is dynamically modeled based on

the mean reprojection error ϵ (in pixels) of the inlier features for that specific motion estimate. The covariance matrix for each visual odometry factor, F_{V_i} , is defined as:

$$F_{V_i} = s \cdot \begin{bmatrix} \epsilon_i^2 & 0 & 0 \\ 0 & \epsilon_i^2 & 0 \\ 0 & 0 & k \cdot \epsilon_i^2 \end{bmatrix} \quad (3.6)$$

where ϵ_i is the mean reprojection error for the i -th camera's visual odometry estimate.

The scaling factor $s = 0.2$ and the orientation variance scaling $k = 0.2$ were determined empirically through trial and error. The robot would be driven in a square (90° turns) and the constants would be adjusted after every run, trying to get as close to the true 90° as possible. This formulation ensures that visual odometry estimates with lower reprojection errors (i.e., sub-pixel accuracy) are considered more reliable. Consequently, the orientation derived from visual odometry, characterized by a variance of $s \cdot k \cdot \epsilon_i^2 = 0.04\epsilon_i^2$, is typically assigned a much lower uncertainty than that from wheel odometry (variance of 0.10), reflecting its higher precision.

4 Results and Analysis

This chapter presents the experimental results for the methodologies proposed in the previous chapter. The analysis is structured to validate the solutions developed to address the core research questions of this thesis. The following sections will detail the performance of the proposed systems, beginning with an evaluation of the RT-SpecFilter's ability to identify and correct specular artifacts in 3D point clouds (Section 4.1). Subsequently, the results from the comparative analysis of feature detectors in specular environments are presented (Section 4.2). The chapter concludes with the experimental validation of the Multicam SP-VO system (Section 4.3), assessing its effectiveness in reducing localization drift.

4.1 RT-SpecFilter Validation Experiments

The ground truth for the following experiments was done by applying the specularity mask on the RGB image, same one detailed in Section 3.2. By comparing the mask with the segmented point cloud, each segmented artifact was manually labeled. An image is labeled as **Specular** when the segmented artifact corresponds to the identified specular region on the RGB image. If the segmentation is triggered but no specularity is identified on the mask, the image is labeled as **Non-Specular**.

4.1.1 Gray Floor with Sun Reflections

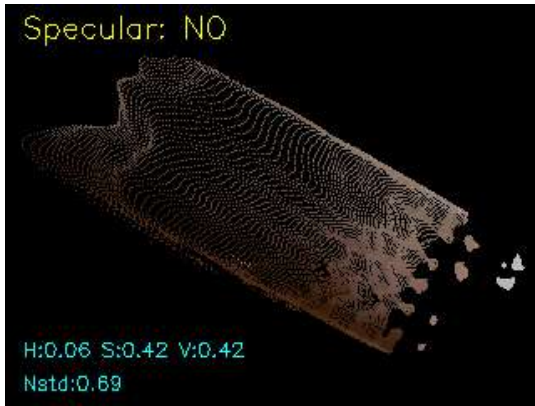
The following testing environment is similar to the actual environment where the Virofighter operates with one minor difference: there are stairs present. The mobile robot will go around the space where several sun reflections are present from nearby windows. In this case, the location of the specular reflection do not seem to move. As discussed in the Theoretical Framework, the position of the specular reflection depends on the po-



(a) Segmented PointCloud with Artifact points projected into an image plane (True Positive)



(b) RGB Image of the sensor looking at highly specular reflection (True Positive)



(c) Segmented PointCloud with Artifact points projected into an image plane (True Negative)



(d) RGB Image of the sensor looking at stairs (True Negative)

Figure 4.1: 3D Pointcloud Image and RGB Image from Pointclouds that triggered the RT-SpecFilter Segmentation Algorithm during the *Gray Floor with Sun Reflections* experiment. First Pair of images(a,b) shows a specularity, while the second pair (c,d) shows the stairs.

sition of the observer and the light source. This scenario presents reflections where the light source (the sun) is not directly reflected, but its light passes through the window. This scenario then presents static specular reflections and stairs. After the run, the correct flattening of the highly specular reflections and the correct classification of the stairs will be observed.

Experiment Results

Table 4.1: Performance Metrics for Polished Floors

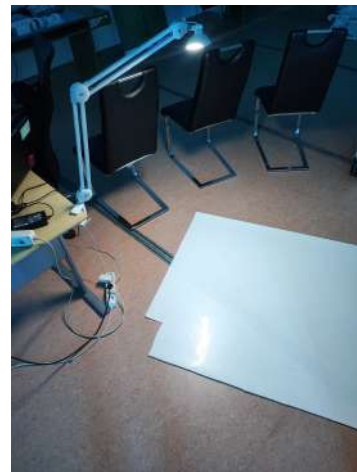
Environment	Accuracy	Precision	Recall	F1 Score
With UVC	97.06%	95.78%	98.02%	0.9803
Without UVC	95.08%	96.89%	97.50%	0.9720

During this experiment, images that triggered the segmentation algorithm were recorded. Figure 4.1 shows examples of these triggers. A true positive case is shown where a 2D projection of the 3D specular artifact (Figure 4.1a) and the corresponding image of the reflection (Figure 4.1b) were correctly identified. In contrast, a true negative case is presented in Figures 4.1c and 4.1d, where the segmentation algorithm was triggered by stairs, but the SVM correctly did not classify the resulting hole as a specularity. This environment was covered both with the UVC lights on and off. Upon further examination of the features that triggered the classifications, the UVC light on the robot increased the amount of specular reflection on the surfaces, leading to more confident classifications. This can be observed on the slight increase in performance in Table 4.1.

4.1.2 Indoors with strong overhead lights



(a) Test Environment in No Overhead Light Scenario with the robot's UVC lights on.



(b) White light over white reflective surface.

Figure 4.2: Indoors with strong overhead lights test environment.

This experiment tests the algorithm's performance in an environment with strong overhead lighting and a highly reflective white surface illuminated by a bright, constant light source. Unlike previous tests, this area contains no stairs. Images of the testing environment as well as the white surface can be seen on Figure 4.2

The robot will navigate this environment while continuously filtering its point cloud under several lighting conditions: no overhead light, low overhead light, and a well-lit setting, each tested with UVC lights on and off. In addition to analyzing the filter's effectiveness, the algorithm will be integrated into the existing exploration implementation

to evaluate its real-world application. The successful completion of the exploration task will be reported as a key performance metric.

Experiment Results

This section provides a detailed analysis of the performance of the RT-SpecFilter algorithm, using data from experiments conducted in an indoor setting with strong overhead lighting and a reflective white surface. The results are presented in Table 4.2 (Performance Metrics for Polished Floors) and Table 4.3 (Confusion matrices for specular classification on polished floors).

Table 4.2: Performance Metrics for Polished Floors

Environment	Accuracy	Precision	Recall	F1 Score	Complete Disinfection
No light w/UVC	90.57%	83.67%	95.35%	0.8913	YES
Low Light w/UVC	94.74%	93.88%	100%	0.9684	YES
Well Lit w/UVC	89.47%	85.15%	95.56%	0.9005	YES
No light	78.63%	73.80%	97.18%	0.8389	YES
Low Light	67.07%	63.50%	94.57%	0.7598	YES
Well Lit	87.69%	90.00%	95.94%	0.9287	YES

Table 4.2 evaluates the overall effectiveness of the system across various lighting conditions, with and without the robot’s UVC lights activated. The metrics—Accuracy, Precision, Recall, and F1 Score—assess the algorithm’s ability to correctly identify and classify specular reflections. A key finding is that the system enabled the robot to achieve "Complete Disinfection" in all test runs, meaning it successfully navigated and covered the entire area without being impeded by false obstacles created by reflections. This was a significant improvement, as the task was previously impossible.

The performance varies with lighting conditions. For instance, in "Low Light w/UVC," the system achieved a high accuracy of 94.74% and a perfect recall of 100%, indicating it identified all true specular reflections. The performance in poor lighting without UVC, such as in the "Low Light" scenario (67.07% accuracy), suggests the underlying SVM classifier model could be improved. However, the implementation includes a histogram-based mapping system that tracks classifications over time. This feature ensures that even if a single classification is incorrect, the system can correct itself, which is why the truly specular areas were consistently filtered out, allowing the robot to complete its task.

The activation of the UVC lights generally improved performance. This is because the lights increased the amount of specular reflection on the floor, making the reflections more pronounced and easier for the system to classify confidently. This is reflected in the higher performance metrics in the "w/UVC" conditions compared to their "no UVC" counterparts.

Table 4.3: Confusion matrices for specular classification on polished floors, showing Actual vs. Predicted counts for each lighting condition.

		Predicted			
		UVC On		No UVC	
Condition	Actual	Specular	Non-Spec.	Specular	Non-Spec.
No Light	Specular	41	2	138	4
	Non-Specular	8	55	49	57
Low Light	Specular	138	0	87	5
	Non-Specular	9	24	50	25
Well Lit	Specular	172	8	378	42
	Non-Specular	30	151	16	35

Table 4.3 provides a closer look at the classifier's performance by showing the confusion matrices for each lighting condition. The following conclusions can be extracted from them:

- **Effect of UVC Light:** In the "No Light" condition with UVC on, there were 41 true positive specular classifications and only 2 false negatives. Without UVC, the number of true positives jumped to 138, but the false negatives remained low at 4. This demonstrates the UVC light's role in creating more confident classifications.
- **Impact of Ambient Light:** In "Well Lit" conditions, the segmentation algorithm is triggered more frequently because the strong ambient light creates a more significant specular component on the reflective floor. This is evident in the high number of classifications made in the "Well Lit" scenarios compared to "No Light" or "Low Light." For instance, with UVC on, there were 172 true positive specular detections in the "Well Lit" case versus 41 in "No Light".
- **System Robustness:** While there are misclassifications (false positives and false

negatives), the overall success of the robot in completing its navigation tasks highlights the robustness of the entire implementation. The histogram feature, which requires an artifact to be consistently classified multiple times before being added to the permanent map, plays a crucial role in mitigating the impact of occasional classifier errors.

4.2 Feature Detector Comparison for Highly Specular Environments

The following analysis details the performance evaluation of four keypoint detectors. The test was conducted on two sequences from the *3DRef: 3D Dataset and Benchmark for Reflection Detection in RGB* [2]. Sequence 1 contains 570 images, while the more extensive Sequence 2 contains 1576 images. Since the dataset does not provide ground truth masks for specular regions, the evaluation utilized the masking algorithm previously mentioned in this thesis to identify and analyze keypoint distribution.

The images in both test sequences have a resolution of 3072x3072 pixels. This high resolution presents a demanding scenario for evaluating both the accuracy and computational efficiency of the detectors. The results for each sequence are summarized in Table 4.4 and Table 4.5.

Table 4.4: Keypoint Detector Performance Summary (Sequence 1 - 570 Images)

Detector	Avg. Specular Keypoint Percentage	Avg. Detection Time (s)
ORB	38.69%	0.057
AKAZE	22.52%	1.160
SIFT	32.22%	2.090
SuperPoint	15.36%	0.810

Table 4.5: Keypoint Detector Performance Summary (Sequence 2 - 1576 Images)

Detector	Avg. Specular Keypoint Percentage	Avg. Detection Time (s)
ORB	36.2%	0.055
AKAZE	22.1%	1.100
SIFT	31.1%	2.087
SuperPoint	17.0%	0.806

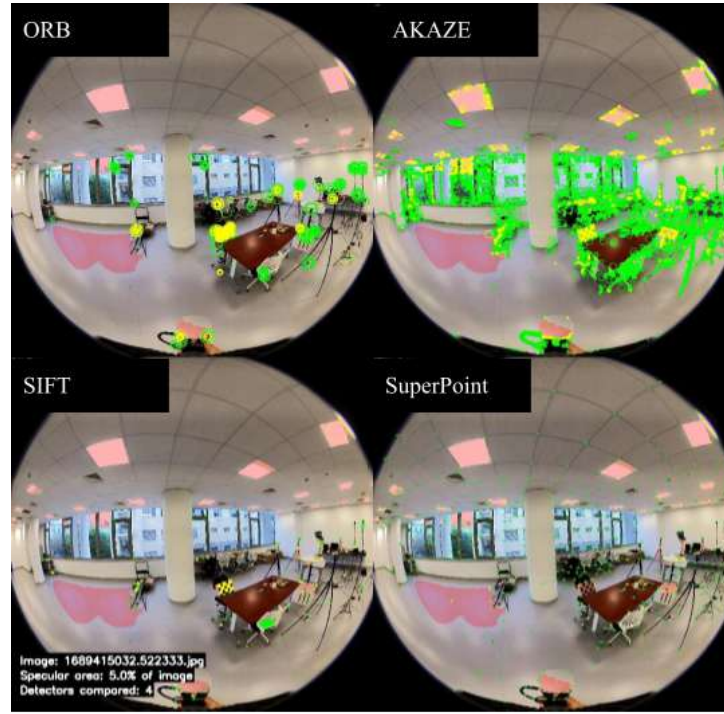


Figure 4.3: Sample Analysis Image from Sequence 2 of the 3DRef: 3D Dataset and Benchmark for Reflection Detection in RGB [2]. Specular Regions are Highlighted in red. Features are colored in green and features detected within Specular Regions are colored in yellow.

Analysis of Results

The results from both sequences are remarkably consistent and reveal a clear trade-off between detection speed, robustness to specular reflections, and the underlying algorithm for each detector.

The classical detectors, **ORB** and **SIFT**, while offering different speed profiles, consistently identified a high percentage of keypoints within specular areas. In Sequence 1, they registered 38.69% and 32.22% respectively, with similar results in Sequence 2 at 36.2% and 31.1%. This consistent behavior makes them less reliable for visual odometry in environments with reflective surfaces. **AKAZE** performed better in terms of specular robustness across both sequences (22.52% and 22.1%) but was significantly slower than ORB.

The **SuperPoint** detector, a deep learning-based model, emerged as the strongest candidate across both tests. It demonstrated the highest robustness by a significant margin, with only 15.36% of its keypoints in specular regions in Sequence 1 and 17.0% in the

much longer Sequence 2. This consistent performance over more than 2,000 combined images is critical for maintaining stable tracking in challenging, real-world conditions.

Furthermore, it is important to analyze the detection time in the context of image resolution. The average detection time for SuperPoint on the high-resolution 3072x3072 test images was stable at approximately 0.81 seconds. While slower than ORB in this specific test, this compute time is a direct consequence of the large number of pixels being processed. For the target robotic platform, which utilizes a 680x460 image resolution, the inference time for the SuperPoint model is substantially faster (capable of running at over 30 Hz), aligning with the requirements for a real-time visual odometry system.

Given its consistent and superior performance in rejecting specular highlights across two extensive test sequences, a critical factor for robust visual odometry, and its high speed at the operational resolution of the real robot, the **SuperPoint** detector is selected as the feature detector for the multicam visual odometry implementation.

4.3 Multicam SP-VO Validation Experiments

To validate the proposed visual odometry using a multi camera wide FoV system alongside the Deep Learning Super Point Model, the following two experiments were conducted.

4.3.1 Visual Validation for Multicam SP-VO

The first experiment is the simplest and serves as quick visual validation. The virofighter mobile robot has an intelisense D435 RGB-D Sensor which is able to create 3D point clouds of its environment. By leveraging the visualization capabilities of the RViz Visualizer for ROS Noetic, the 3D pointclouds will be projected onto the world plane according to the robot's world pose at the time of the capture.

The environment to capture is the one shown in Figure 4.4a. The robot will go around once, capturing the 3d information of the floor and edges of the environment. The result of the proposed multi camera system will be compared to the current implementation that uses only the wheel odometry as means of localization. The expected result is for the shape of the box arrangement to visibly drift when using only the wheel odometry in

comparison of when the multi camera system is used. As expected, the shape of the box arrangement showed more visible drift when using only wheel odometry (Figure 4.5a) compared to when using the multi-camera system (Figure 4.5b).

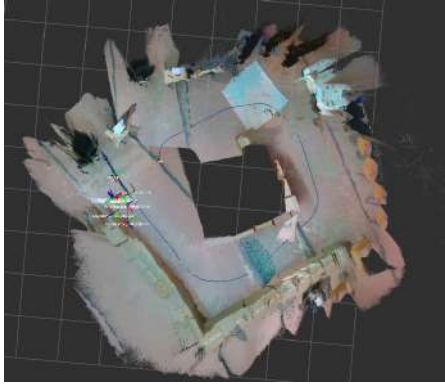


(a) Environment for visual validation of Multicam SP-VO.

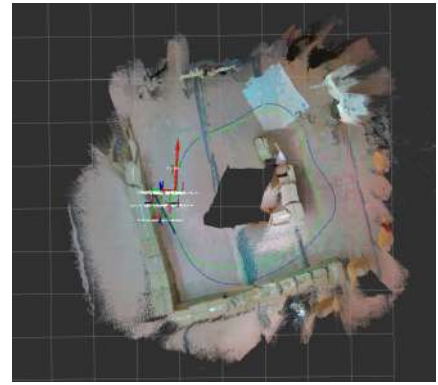


(b) Environment for measuring landmark drift using ArUco markers.

Figure 4.4: Experimental setups for Visual Odometry validation, illustrating the environments used for direct visual validation and landmark-based drift assessment.



(a) 3D reconstruction using Wheel Odometry as reference



(b) 3D reconstruction using Multicam Visual Odometry as reference

Figure 4.5: Comparison between 3D reconstruction of the test environment using a D435 Intel Realsense Camera

4.3.2 Landmark Drift as Validation for Multicam SP-VO

To validate the performance of the Multicam SP-VO system, this experiment uses four ArUco [42] markers as fixed landmarks within the environment (Figure 4.4b). These markers were selected thanks to the pose estimation capabilities, allowing the robot not only to estimate their position, but also orientation. The robot repeatedly navigated the area, and by comparing the pose estimates of each marker over time against their initial observation, the system's cumulative drift in position (x, y) and orientation (yaw) was quantified.

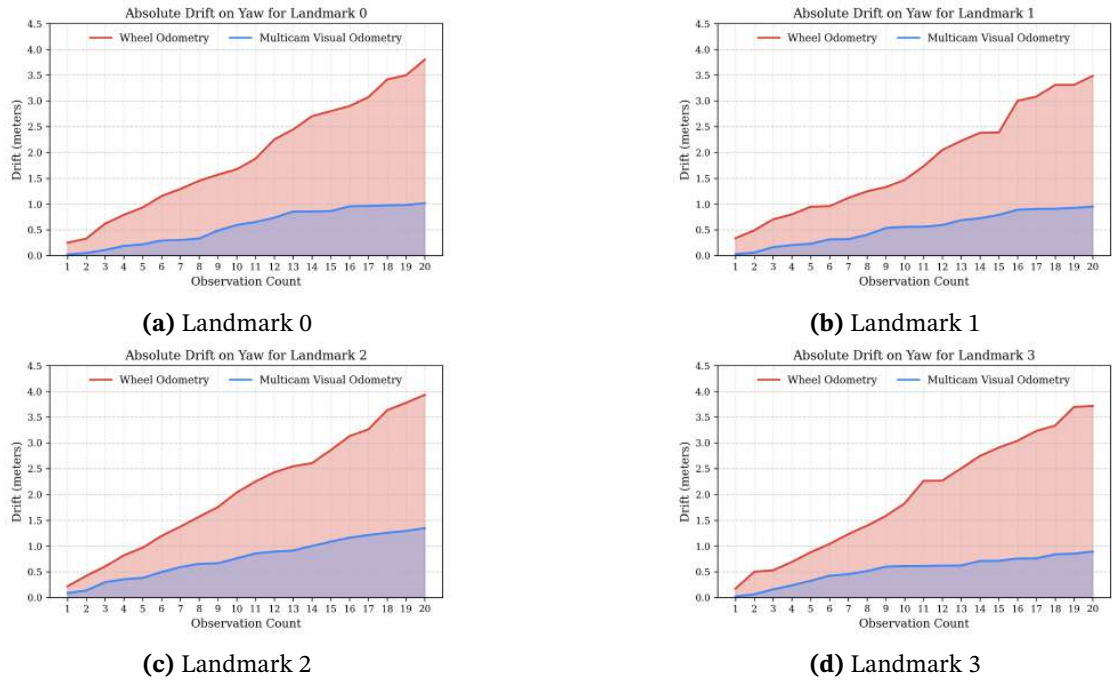


Figure 4.6: Absolute Drift Comparison between the Wheel Odometry and the Proposed Multicam Visual Odometry on Yaw

The primary hypothesis is that, while positional drift is expected—since the system’s scale is derived from wheel odometry, the fusion of multi-camera data should drastically reduce orientation drift. This significant improvement in yaw stability is confirmed in Figure 4.6, which plots the accumulated absolute yaw drift for each system. The proposed multi-camera approach demonstrates substantially lower orientation error compared to using wheel odometry alone. A comprehensive comparison of the absolute drift for all axes can be found in Annex B. This data was collected over a 20-minute run, with approximately 60 seconds between consecutive observations of the same landmark.

An analysis of the positional drift is presented in Figure 4.7. Although the multi-camera system does not eliminate positional drift, it maintains a highly consistent estimate of the markers’ orientation over time. As a result, the positional drift, shown on the right side of the figure, accumulates systematically in a single, predictable direction rather than wandering randomly. This contrasts with pure wheel odometry, where compounding orientation errors lead to more erratic positional drift. The trajectory of Landmark 2 (Figure 4.7d), which is completely opposite to the origin (landmark 0), shows the most variance in drift since the errors tend to cancel out when performing a loop.

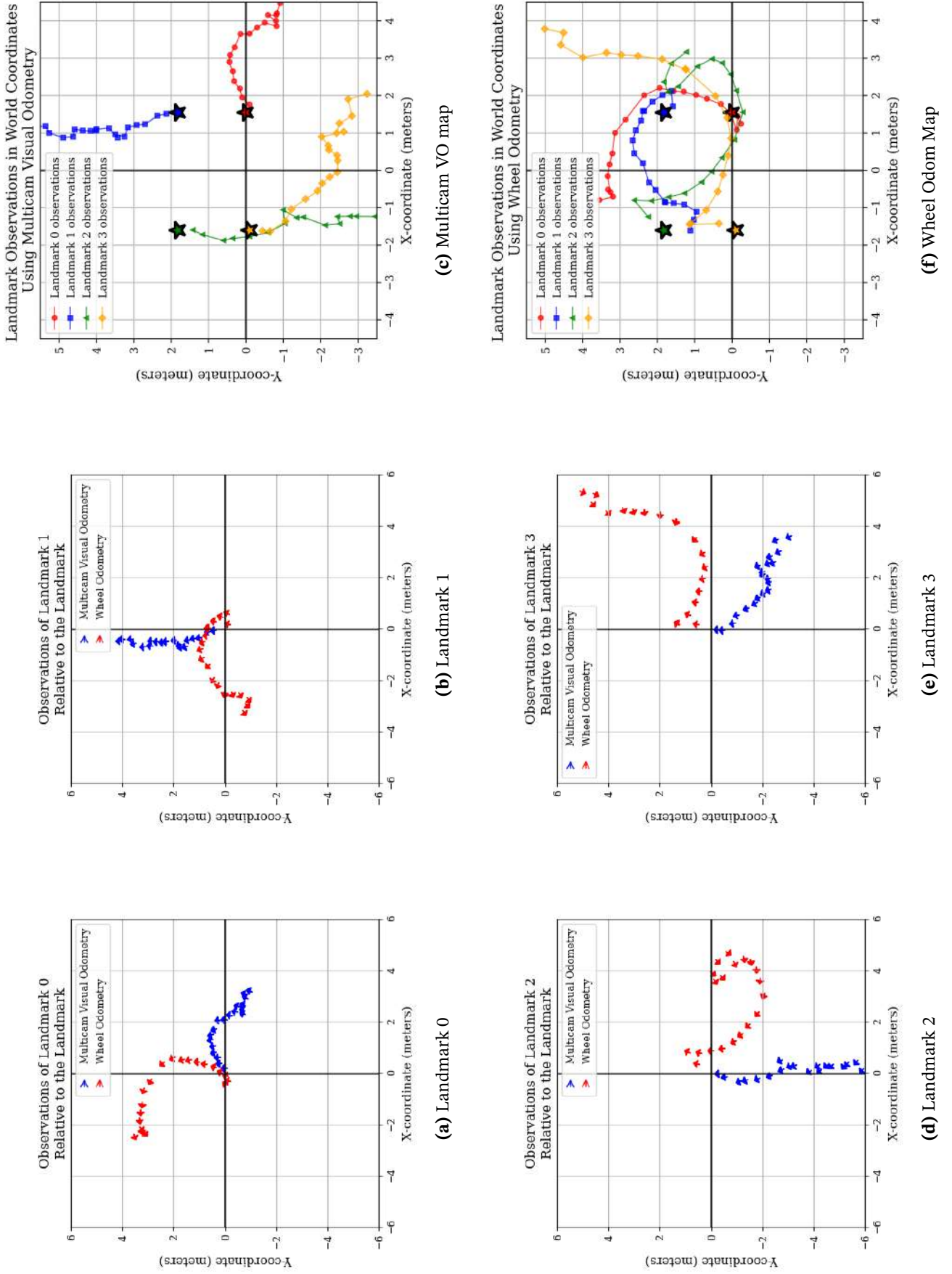


Figure 4.7: Pose Drift Comparison between observations (Landscape, 3x2)

5 Conclusion and Future Work

This thesis tackled the challenge of achieving robust indoor mapping and localization for autonomous robots in environments dominated by specular reflections. Such reflections, prevalent on shiny surfaces, introduce significant noise into sensor data, making it harder for onboard SLAM systems to work. The research focused on three primary objectives: mitigating the impact of specular reflections on RGB-D camera data, assessing the robustness of feature detectors in specular-rich settings, and developing a multi-camera visual odometry system to enhance localization precision.

To address these objectives, the RT-SpecFilter was introduced. A real-time, dual-threaded algorithm tailored to identify and correct specular artifacts in point clouds generated by the Intel RealSense D435 camera. Experimental evaluations highlighted the algorithm's effectiveness, even with the onboard UVC lights on.

Additionally, a detailed comparison of feature detectors SIFT, ORB, AKAZE, and SuperPoint was performed using datasets characterized by high specularity. The results revealed that classical detectors, such as ORB and SIFT, frequently detected keypoints in unreliable specular regions, whereas the deep-learning-based SuperPoint detector exhibited superior robustness, consistently minimizing keypoint detection in these areas. This resilience, combined with its real-time processing capabilities, positioned SuperPoint as the optimal choice for the visual odometry framework developed in this work.

Leveraging these insights, a multi-camera visual odometry system was designed, using four wide FoV cameras and the SuperPoint model, thus creating the Multicam SP-VO system. By integrating visual odometry estimates with wheel odometry through a pose-graph optimization framework, the system achieved a noticeable reduction in orientation drift compared to wheel odometry alone. Although positional drift persists due to

the reliance on wheel odometry for scale, the system’s consistent orientation estimates ensured that drift accumulated predictably along a single trajectory, a significant advancement over the erratic drift observed with standalone wheel odometry.

5.1 Limitations and Future Work

Despite the success of RT-SpecFilter, its performance could be further refined in environments where specular reflections and physical holes frequently coexist. The current approach relies on a depth-based segmentation prior to classification, which may conflate multiple artifacts. A promising enhancement would involve incorporating a clustering mechanism post-segmentation to treat distinct artifacts independently, thereby improving classification granularity and accuracy.

The SuperPoint model’s robust performance highlights the potential of deep learning in managing specular environments. However, as a pre-trained model, it was not specifically optimized for specular surfaces. Future efforts could focus on fine-tuning SuperPoint using a curated dataset of specular-rich images to further suppress keypoint detection in reflective regions. This could be achieved by designing a convolutional neural network or a transformer layer head to explicitly penalize specular keypoint detection.

The multi-camera visual odometry system, while effective with a 2D-2D motion estimation approach, faced challenges with a 3D-2D (PnP) method due to limited stereo correspondences. Future research could explore direct visual odometry techniques, which leverage pixel intensities to generate dense 3D reconstructions, potentially enabling a robust PnP solution. Additionally, integrating advanced deep learning models for monocular depth estimation, such as Meta’s DinoV2, could provide accurate 3D data from wide FoV cameras, eliminating scale ambiguity and reducing dependence on wheel odometry.

In summary, this thesis presents a cohesive solution for enhancing robotic navigation in specular-rich indoor environments through targeted artifact filtering, robust feature detection, and multi-sensor fusion. The results affirm that systematically addressing specular reflection challenges significantly improves the accuracy and reliability of autonomous systems, fostering their practical deployment in demanding settings such as hospitals and industrial facilities.

References

- [1] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010. <https://doi.org/10.1109/MITS.2010.939925>
- [2] X. Zhao and S. Schwertfeger, “3dref: 3d dataset and benchmark for reflection detection in rgb and lidar data,” *3DV*, 2024.
- [3] S. Zhang, “Diffuse structured light,” *Optics and Lasers in Engineering*, vol. 66, pp. 210–215, 2015. <https://doi.org/10.1016/j.optlaseng.2014.09.011>
- [4] A. Al-Baghdadi, O. Mazahreh, A. Al-Bashir, M. Fraiwan, and L. Abualigah, “Improving 3D reconstruction through RGB-D sensor noise modeling,” *Sensors*, vol. 24, no. 3, p. 950, 2024. <https://doi.org/10.3390/s24030950>
- [5] A. Anwer, S. Ainouz, M. N. M. Saad, S. S. Ali, and F. Meriaudeau, “SpecSeg network for specular highlight detection and segmentation in real-world images,” *Sensors*, vol. 22, no. 17, p. 6552, Aug 2022. <https://doi.org/10.3390/s22176552>
- [6] D. DeTone, T. Malisiewicz, and A. Rabinovich, “Superpoint: Self-supervised interest point detection and description,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236. <https://doi.org/10.1109/CVPRW.2018.00060>
- [7] S. A. Shafer, “Using color to separate reflection components,” *Color Research & Application*, vol. 10, no. 4, pp. 210–218, 1985. <https://doi.org/10.1002/col.5080100409>

- [8] M. D. Santambrogio, D. Sciuto, P. M. R. S. Reis, and J. M. F. Moura, Eds., *Time-of-Flight and Structured Light Depth Cameras: Technology and Applications*. Cham: Springer International Publishing, 2022. <https://doi.org/10.1007/978-3-031-06687-6>
- [9] D. Scaramuzza *et al.*, “A flexible technique for accurate omnidirectional camera calibration and structure from motion,” in *ICVS*, 2006.
- [10] H. P. Moravec, “Towards automatic visual obstacle avoidance,” in *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 2*, 1979, pp. 778–784. <https://doi.org/10.5555/1624764.1624893>
- [11] D. Nistér, “Visual odometry,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 652–659. <https://doi.org/10.1109/CVPR.2004.1315134>
- [12] D. Scaramuzza and F. Fraundorfer, “Visual odometry: Part i - the first 30 years and fundamentals,” *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.
- [13] D. Nistér, “An efficient solution to the five-point relative pose problem. ieee-tpami 26(6), 756-770,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, pp. 756–77, 07 2004. <https://doi.org/10.1109/TPAMI.2004.17>
- [14] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [15] F. Fraundorfer and D. Scaramuzza, “Visual odometry: Part ii - matching, robustness, optimization, and applications,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 1, pp. 78–90, 2012.
- [16] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, 2010. <https://doi.org/10.1109/MITS.2010.938928>

- [17] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “Pose graph optimization for robust slam,” *IEEE Transactions on Robotics*, 2011. <https://doi.org/10.1109/TRO.2011.2160467>
- [18] F. Dellaert, “Factor graphs and gtsam: A hands-on introduction,” Georgia Institute of Technology, Tech. Rep., 2012. [Online]. Available: https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam_tutorial.pdf
- [19] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “isam2: Incremental smoothing and mapping using the bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.
- [20] A. Aladrén, G. López-Nicolás, L. Puig, and J. J. Guerrero, “Navigation assistance for the visually impaired using rgb-d sensor with range expansion,” *IEEE Systems Journal*, vol. 10, no. 3, pp. 922–932, 2016. <https://doi.org/10.1109/JSYST.2014.2320639>
- [21] A. Morgand and M. Tamaazousti, “Generic and real-time detection of specular reflections in images,” in *Proc. Int. Conf. Comput. Vis. Theory Appl. (VISAPP)*, Lisbon, Portugal, 2014, pp. 274–282. <https://doi.org/10.5220/0004714902740282>
- [22] W. Feng, J. Sun, Q. Liu, X. Li, D. Liu, and Z. Zhai, “Specular highlight removal of light field image combining dichromatic reflection with exemplar patch filling,” *Optics and Lasers in Engineering*, vol. 178, p. 108175, Jul 2024. <https://doi.org/10.1016/j.optlaseng.2024.108175>
- [23] S. Jo, O. Jang, C. Bhattacharyya, M. Kim, T. Lee, Y. Jang, H. Song, H. Kwon, S. Do, and S. Kim, “S-LIGHT: Synthetic dataset for the separation of diffuse and specular reflection images,” *Sensors (Basel, Switzerland)*, vol. 24, no. 7, p. 2286, 2024. <https://doi.org/10.3390/s24072286>
- [24] R. Or-El, R. Hershkovitz, A. Wetzler, G. Rosman, A. M. Bruckstein, and R. Kimmel, “Real-time depth refinement for specular objects,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 4378–4386. <https://doi.org/10.1109/CVPR.2016.475>

- [25] T. Yoshida, V. Golyanik, O. Wasenmüller, and D. Stricker, “Improving time-of-flight sensor for specular surfaces with shape from polarization,” in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Athens, Greece, 2018, pp. 1558–1562. <https://doi.org/10.1109/ICIP.2018.8451357>
- [26] R. Koch and M. Kühn, *Detection of Specular Reflections in Range Measurements for Faultless Robotic SLAM*. Cham, Switzerland: Springer International Publishing, 2016, ch. 11, pp. 313–343. https://doi.org/10.1007/978-3-319-27146-0_11
- [27] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004. <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- [28] S. A. K. Tareen and Z. Saleem, “A comparative analysis of sift, surf, kaze, akaze, orb, and brisk,” *IEEE International Conference on Computing, Mathematics and Engineering Technologies*, pp. 1–10, 2018. <https://doi.org/10.1109/ICOMET.2018.8346440>
- [29] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571. <https://doi.org/10.1109/ICCV.2011.6126544>
- [30] P. F. Alcantarilla and T. Solutions, “Fast explicit diffusion for accelerated features in nonlinear scale spaces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2013. <https://doi.org/10.5244/C.26.13>
- [31] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, “Kaze features,” in *European Conference on Computer Vision*, 2012, pp. 214–227. https://doi.org/10.1007/978-3-642-33783-3_16
- [32] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect visual odometry for monocular and multicamera systems,” *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, Apr 2017. <https://doi.org/10.1109/TRO.2016.2623335>
- [33] K. Tateno, F. Tombari, I. Armeni, and N. Navab, “CNN-SLAM: Real-time

- dense monocular SLAM with learned depth prediction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6572–6580. <https://doi.org/10.1109/CVPR.2017.695>
- [34] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers, “Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [35] N. Yang, L. von Stumberg, R. Wang, and D. Cremers, “D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [36] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *European conference on computer vision*. Springer, 2014, pp. 834–849.
- [37] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, “Benefit of large field-of-view cameras for visual odometry,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016, pp. 801–808. <https://doi.org/10.1109/ICRA.2016.7487210>
- [38] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” in *Lecture Notes in Computer Science*, vol. 2049, 09 2001, pp. 249–257. https://doi.org/10.1007/3-540-44673-7_12
- [39] S. Kim, M. Ra, and W.-Y. Kim, “Specular detection on glossy surface using geometric characteristics of specularly in top-view images,” *Sensors*, vol. 21, p. 2079, 03 2021. <https://doi.org/10.3390/s21062079>
- [40] J. Maye, P. Furgale, and R. Siegwart, “Self-supervised calibration for robotic systems,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 473–480. <https://doi.org/10.1109/IVS.2013.6629513>
- [41] H. Seok and J. Lim, “ROVO: Robust omnidirectional visual odometry for wide-baseline wide-FOV camera systems,” in *2019 International Conference on*

Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 6344–6350.

<https://doi.org/10.1109/ICRA.2019.8793758>

- [42] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, p. 2280–2292, 06 2014.
<https://doi.org/10.1016/j.patcog.2014.01.005>

Abstract

Enhancing Indoor Mapping and Localization in Specular Rich Environments using Deep Learning and Sensor Fusion

Renatto Tommasi Hernández

Robotic indoor mapping and localization are significantly challenged in environments with highly reflective or specular surfaces, which are common in hospitals and industrial settings. Specular reflections introduce severe artifacts in depth data from RGB-D sensors and degrade the performance of visual Simultaneous Localization and Mapping (SLAM) systems by creating unreliable features. This thesis presents a comprehensive solution to enhance robotic navigation in such specular-rich environments through a combination of deep learning and multi-sensor fusion. We propose a real-time filtering algorithm, RT-SpecFilter, which uses a Support Vector Machine (SVM) to detect and mitigate specular artifacts in point clouds from an Intel RealSense D435 camera. Furthermore, we conduct a comparative analysis of feature detectors, identifying SuperPoint as the most robust for environments with specular highlights. Finally, we develop the Multicam SP-VO system that leverages four wide FoV cameras and fuses their motion estimates with wheel odometry data using a pose-graph optimization framework. Experimental results demonstrate that the proposed system significantly reduces orientation drift improves localization accuracy compared to reliance on wheel odometry alone and mitigates the specular artifacts during mapping, thereby enabling more robust and reliable autonomous navigation in challenging indoor spaces.

Keywords: Indoor Localization, SLAM, Specular Reflections, Sensor Fusion, Visual Odometry, Deep Learning, Multi-Camera Systems

Sažetak

Unaprjeđenje izrade karte i lokalizacije u zatvorenom prostoru s izraženim zrcalnim odrazima pomoću dubokog učenja i fuzije senzora

Renatto Tommasi Hernández

Robotska lokalizacija i mapiranje u unutarnjim prostorima otežani su zbog zrcalnih površina koje uzrokuju artefakte u podacima s RGB-D senzora i narušavaju rad SLAM sustava. Ovaj rad predstavlja rješenje koje kombinira duboko učenje i fuziju senzora za poboljšanje robotske navigacije. Predložen je filter u stvarnom vremenu (RT-SpecFilter) zasnovan na SVM-u za uklanjanje zrcalnih artefakata detektiranih Intel RealSense kamerom. Nakon usporedne analize, SuperPoint je odabran kao najrobusniji detektor značajki. Na temelju toga, razvijen je višekamerni sustav vizualne odometrije koji spaja podatke s četiri 'fisheye' kamere i odometrije kotača pomoću optimizacije grafa poza. Eksperimentalni rezultati potvrđuju da predloženi sustav značajno smanjuje odstupanje u orijentaciji (drift) i poboljšava točnost lokalizacije u usporedbi sa samom odometrijom kotača, omogućujući time robusniju i pouzdaniju autonomnu navigaciju.

Ključne riječi: Unutarnja lokalizacija, SLAM, Zrcalne refleksije, Fuzija senzora, Vizualna odometrija, Duboko učenje, Višekamerni sustavi

Annex A: Camera Calibration Parameters

Table 1: Camera Parameters

Parameter	cam0	cam1	cam2	cam3	cam4
ROS Topic	/d435/color /image_raw	/fisheye_front /image_raw	/fisheye_right /image_raw	/fisheye_rear /image_raw	/fisheye_left /image_raw
Camera Model	pinhole	pinhole	pinhole	pinhole	pinhole
Distortion Model	radtan	equidistant	equidistant	equidistant	equidistant
Intrinsic Params					
f_x	611.36	271.71	329.70	271.44	274.46
f_y	610.25	272.14	341.08	271.57	275.01
c_x	326.35	320.79	317.70	316.00	329.30
c_y	246.58	226.65	217.58	239.38	227.55
Distortion Coeffs					
d_0	0.19	-0.05	0.05	-0.05	-0.06
d_1	-0.38	0.00	-0.13	0.00	0.01
d_2	0.01	-0.00	-0.06	-0.00	-0.01
d_3	-0.00	0.00	0.06	0.00	0.00
Resolution (w, h)	[640, 480] px	[640, 480] px	[640, 480] px	[640, 480] px	[640, 480] px
Overlapping Cams	[1]	[0, 2, 4]	[1, 3]	[2, 4]	[1, 3]
Transformation	$T_{\text{base\_to\_cam0}} =$ $\begin{bmatrix} 0.51 & 0.00 & 0.86 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 \\ -0.86 & 0.00 & 0.51 & 1.81 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$	$T_{\text{cam1\_cam0}} =$ $\begin{bmatrix} 1.00 & 0.02 & -0.00 & -0.03 \\ -0.01 & 0.89 & 0.45 & 0.08 \\ 0.01 & -0.45 & 0.89 & 0.02 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$	$T_{\text{cam2\_cam1}} =$ $\begin{bmatrix} 0.03 & 0.57 & -0.82 & -0.26 \\ -0.63 & 0.65 & 0.43 & 0.13 \\ 0.77 & 0.51 & 0.38 & -0.12 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$	$T_{\text{cam3\_cam2}} =$ $\begin{bmatrix} -0.02 & 0.63 & -0.78 & -0.17 \\ -0.47 & 0.68 & 0.56 & 0.15 \\ 0.88 & 0.38 & 0.28 & -0.26 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$	$T_{\text{cam4\_cam3}} =$ $\begin{bmatrix} 0.01 & 0.51 & -0.86 & -0.29 \\ -0.61 & 0.68 & 0.40 & 0.10 \\ 0.79 & 0.52 & 0.32 & -0.14 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$

Annex B: Landmark Drift Full Results

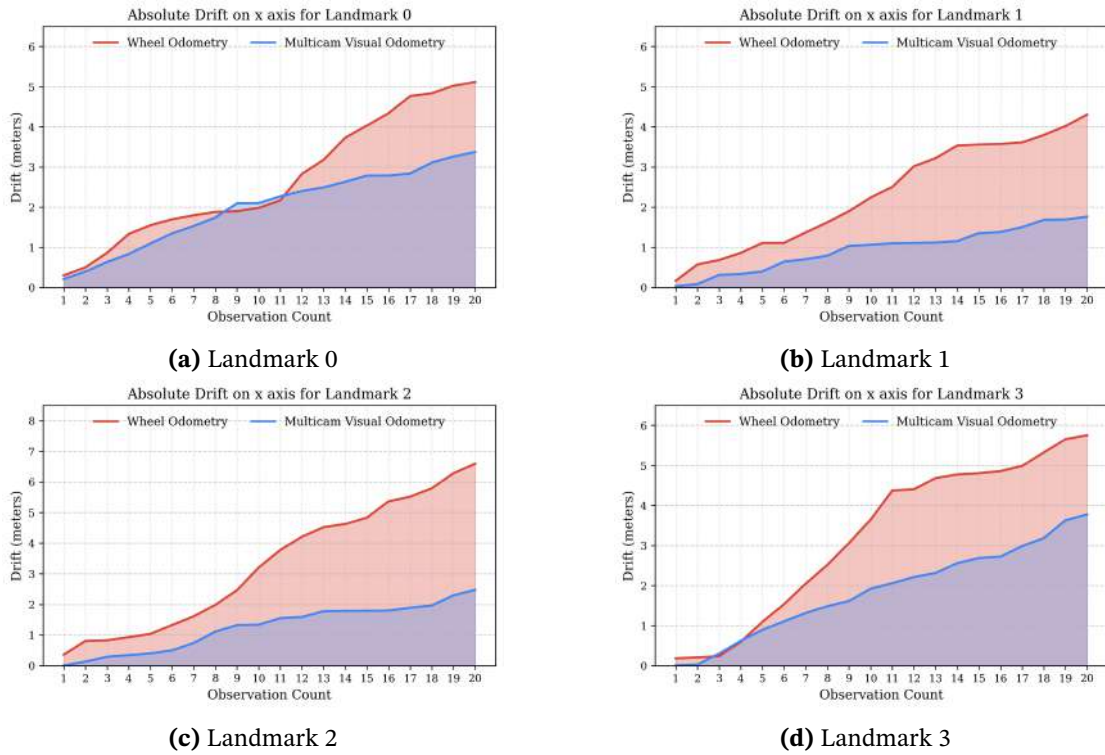


Figure 1: Absolute Drift Comparison between the Wheel Odometry and the Proposed Multicam Visual Odometry along the x axis

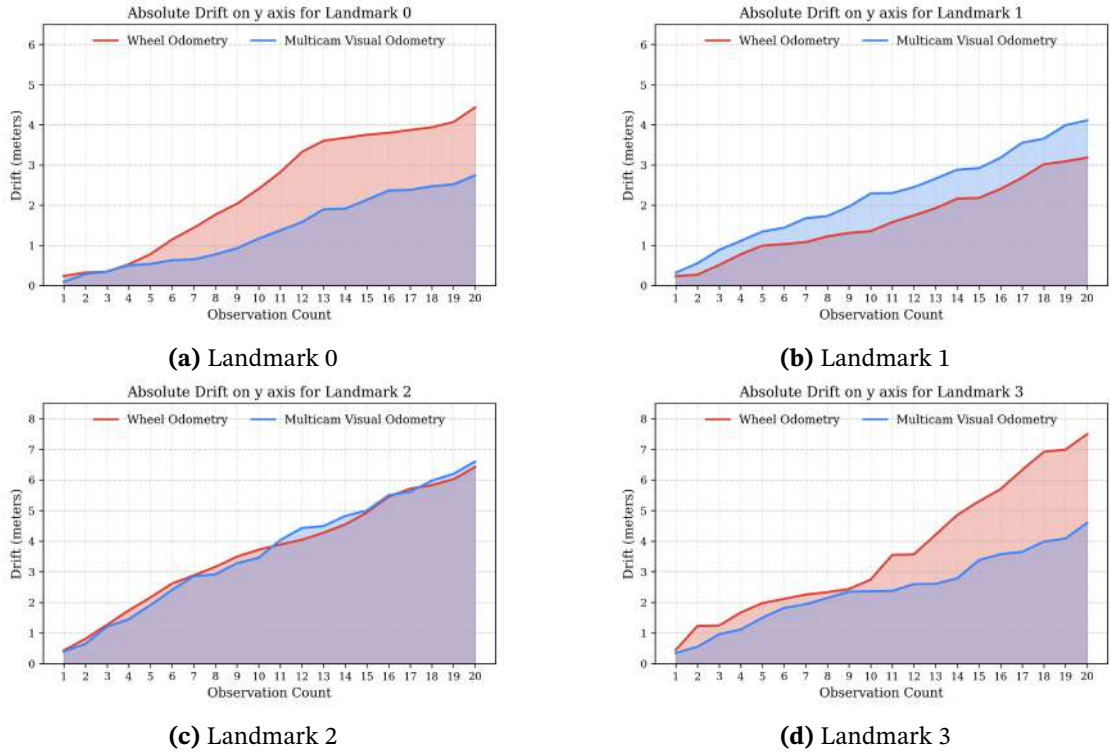


Figure 2: Absolute Drift Comparison between the Wheel Odometry and the Proposed Multicam Visual Odometry along the y axis

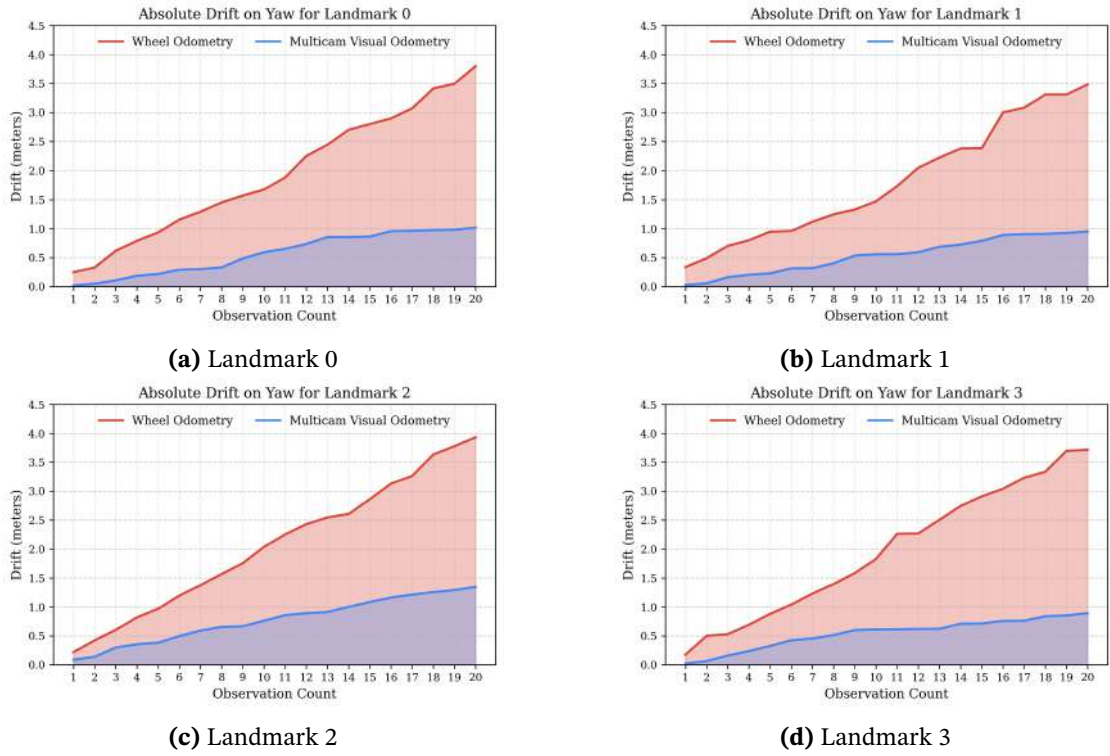


Figure 3: Absolute Drift Comparison between the Wheel Odometry and the Proposed Multicam Visual Odometry on Yaw

11

Detection of Invisible Obstacles for Drone Obstacle Avoidance
Sawera Yaseen

UNIVERSITY OF ZAGREB
**FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING**



MASTER'S THESIS No. 000

Detection of Invisible Obstacles for Drone Obstacle Avoidance

Sawera Yaseen

Zagreb, June 2025.

First and foremost, I would like to express my sincere gratitude to my supervisors, Dr. SC. Stjepan Bogdan and Benjamin van Manen, for their invaluable guidance, encouragement, and continuous support throughout the duration of this work.

Their invaluable feedback and mentorship have been instrumental in the development of this thesis.

My heartfelt thanks go to my professors at the University of Girona and the University of Zagreb for their support and for providing me with the necessary resources to enhance my learning and research. Their guidance and expertise have been invaluable in shaping my academic journey. I am also thankful to my colleagues and friends within the IFROS program who made this program memorable.

I would like to express my deepest gratitude to my family, whose love and support have been the cornerstone of my journey. To my parents, thank you for your endless encouragement, unconditional love, and heartfelt prayers that have guided me through every challenge. To my siblings, thank you for being my constant source of motivation and for always cheering me on, no matter the distance or circumstances.

Abstract

Detection of Invisible Obstacles for Drone Obstacle Avoidance

Autonomous urban drone navigation presents unique challenges due to the presence of power lines, which are often missed by conventional obstacle detection systems. This thesis addresses the paramount issue of power line detection and ranging, thereby enhancing the safety and reliability of autonomous UAV operations. To tackle this challenge, two approaches are proposed and evaluated. The first method involves developing an object detection and ranging pipeline using YOLOv8 for power line detection, in conjunction with stereo vision for distance estimation. The second method involves segmenting power lines employing a UNet-based architecture, followed by monocular depth estimation to infer the power line distance from the camera. Furthermore, the IMU data is utilized to address the scale ambiguity inherent in monocular vision for more accurate distance recovery. Both methodologies have been trained and evaluated utilizing a combination of the publicly available PLDataset, TTPLA datasets, and a custom-annotated dataset. The experimental results demonstrate the effectiveness of both approaches in detecting and ranging the cables under challenging visual conditions. This research contributes to the advancement of robust obstacle avoidance systems, facilitating the safe operation of low-altitude drones within urban environments.

Keywords: Drone Navigation, Obstacle Avoidance, Power Line Detection, Stereo Vision, Monocular Depth Estimation

CONTENTS

List of Figures	vi
List of Tables	vii
1. Introduction	1
1.1. Background and Context	1
1.2. Problem Statement	2
1.3. Research Questions	3
1.4. Objectives and Scope	3
1.5. Thesis Structure	4
2. Literature Review	5
2.1. Traditional Approaches	5
2.2. Deep Learning Techniques	6
2.2.1. Detection-Based Methods	7
2.2.2. Segmentation-Based Methods	8
2.3. Stereo vision based distance estimation	10
2.4. 2D-to-3D Reconstruction of Linear Structures	11
2.5. Research Gaps	13
3. Power line Detection and Distance Estimation using Object Detection and Stereo Vision	15
3.1. Datasets for Object Detection	15
3.1.1. Annotations	16
3.2. Object Detection Models	16
3.2.1. You Only Look Once (YOLO) version 5	17
3.2.2. You Only Look Once (YOLO) version 8	18
3.2.3. You Only Look Once (YOLO) version 11	21
3.3. Model Training	22

3.4. Object Detection Performance Metrics	23
3.5. Distance Estimation	23
3.5.1. Stereo Camera Setup	23
3.5.2. Stereo-Vision and Depth Estimation	25
3.5.3. Power Line Distance Estimation Integration Pipeline . . .	26
3.6. Results	28
4. Semantic Segmentation and 2D-3D Reconstruction of Power Lines	32
4.1. Segmentation Datasets	32
4.1.1. Annotations	32
4.2. Deep Learning based Segmentation Models	33
4.2.1. UNet Architecture	33
4.3. Model Training	39
4.4. Monocular Vision-based Distance Estimation	39
4.4.1. 2D-3D Reconstruction	40
4.5. Integrated Framework for Power Line Distance Estimation	45
4.6. Evaluation Metrics	47
4.7. Results	47
5. Conclusion	54
5.1. Addressing the Research Questions	54
5.2. Recommendations for further research	55
Bibliography	57

LIST OF FIGURES

3.1. Object Detection Models [1]	17
3.2. YOLOv8 Architecture	18
3.3. Schematic Representation of Power Line Distance Estimation . . .	23
3.4. Typical Stereo-vision System	25
3.5. Stereo vision-based distance estimation methodology	27
3.6. Comparision of YOLO Models	29
3.7. Detected power lines with the estimated distance ($GT = 2\text{ m}$) . .	30
3.8. Detected power lines with the estimated distance ($GT = 3.5\text{ m}$) .	30
3.9. Detected power lines with the estimated distance ($GT = 7\text{ m}$) . .	30
3.10. Impact of bounding box variations on distance estimation accuracy	31
4.1. UNet Architecture	34
4.2. Geometric Reconstruction Pipeline	41
4.3. Power Line Distance Estimation Pipeline	45
4.4. U-Net model visualization for Power line segmentation: columns display from left to right: the original image, ground truth masks, and model predictions	49
4.5. Results (Ground Truth 2m)	50
4.5. Results (Ground Truth 2m) (continued)	51
4.6. Results (Ground Truth 3.5m)	51
4.6. Results (Ground Truth 3.5m) (continued)	52
4.7. Results (Ground Truth 7m)	52
4.7. Results (Ground Truth 7m) (continued)	53
4.8. Execution time per frame for each operation in the pipeline	53

LIST OF TABLES

3.1. Training hyperparameters for YOLOv5, YOLOv8, and YOLOv11 models.	22
3.2. Evaluation metrics used for assessing the object detection models.	24
3.3. Comparison of Model Performance Metrics	29
4.1. Evaluation metrics used for assessing distance estimation performance.	47
4.2. UNet Segmentation Model Performance Metrics for Power Lines .	48
4.3. Distance estimation performance at different ground truth distances.	50

1. Introduction

This chapter provides the foundation of the thesis by first giving the background and context on the use of Unmanned Aerial Vehicles (UAVs) and the inherent challenges that arise in executing secure operations within cluttered environments containing barely visible obstacles such as power lines. Based on this background, the chapter then articulates the problem addressed by this research, formulates the guiding research questions, and outlines the objectives and scope of the work. Finally, the chapter concludes by outlining the structure of the remainder of the thesis.

1.1. Background and Context

Unmanned aerial vehicles (UAVs) have experienced significant growth in recent years, with applications spanning surveillance [2], logistics [3], and infrastructure inspection [4]. Industrial UAVs are increasingly utilized for aerial mapping [5], photogrammetry [6] precision agriculture [7], forest fire monitoring [8], and autonomous security, while consumer drones are primarily employed for photography, entertainment, and personal use [9]. They have become a vital tool in modern technology owing to their ability to navigate remote or uncertain environments, enhancing their applicability across diverse domains [10], [9]. This surge is further fueled by advanced autonomy and miniaturization, facilitating more proficient and intelligent operational capabilities. Due to their high mobility and versatility, low-altitude UAVs are widely employed for different applications across a wide range of fields [10].

The extensive utilization of UAVs highlights the necessity of ensuring autonomous and safe navigation across all operational environments. Recent studies accentuate that obstacle avoidance is pivotal for successfully executing UAV missions since static (Trees, buildings) and dynamic (other aircraft) obstacles threaten safe aerial operations. For preventing collisions and operational failures,

sophisticated perception systems must be incorporated into UAV control [9].

A range of sensors can be added to a UAV for environmental perception. Common sensors include LiDAR(Laser-based range scanners), Vision(Monocular, Stereo, RGB-D cameras), infrared (IR), radar(MIMO or conventional) and millimeter-wave sensors. Each sensor offers different advantages and limitations [9].

Within the spectrum of sensing modalities, vision-based methods are popular due to their superior resolution and versatility. A monocular camera, characterized by a single lens, generates a dense two-dimensional image but lacks direct metric depth information. To ascertain the range, monocular systems require motion cues or trained models: methods such as monocular depth estimation or 3D reconstruction methods regress pixel disparities across multiple frames. [11]. Such methods can recover relative depth up to an unknown scale factor. Conversely, Stereo cameras utilize two synchronized lenses to estimate binocular depth by feature matching in left and right images, a disparity map is generated and triangulated to get the metric depth. Thus, stereo vision provides real-world distance measurements without the scale ambiguity inherent in monocular setups. Nevertheless, stereo matching depends on textured scenes and may encounter difficulties in low-light or long-range scenarios. The precision of stereo depth estimation decreases under suboptimal illumination conditions or as distance increases [12]. In conclusion, vision-based system trade-off the requirement for computational depth recovery against compatibility and rich visual representation.

Despite these technological advancements, UAVs flying at low altitudes remain susceptible to significant safety challenges due to hard-to-see obstacles. This has created a substantial need for reliable obstacle avoidance systems, particularly in low-altitude cluttered environments filled with thin, nearly invisible obstacles [13].

1.2. Problem Statement

Aerial slender targets, particularly overhead powerlines, are among the most difficult obstacles for UAVs to detect, their long, slender structure, low-reflectivity, variable appearance under different lighting conditions, and minimal visual contrast against complex backgrounds (tree branches, sky), making them prone to missed detection [14], [15], [16]. The focus of this thesis is on the specific challenge of detecting and estimating the range of power lines for low-altitude UAVs. The aim is to identify the presence of the power lines within the drone's path and accurately calculate their distance to facilitate obstacle avoidance. Power lines

fundamentally differ from conventional targets in object detection as they occupy a minuscule portion of the images (merely several pixels) and often extend across the entire frame. As a result, traditional obstacle detectors frequently fail to detect or accurately segment these objects.

Furthermore, the distance estimation for these obstacles constitutes a pivotal aspect of the problem. Knowing the position of a power line relative to the UAV is essential for collision avoidance. Thus, this research challenge encompasses not only the detection of the power line in the image but also the determination of its distance within three-dimensional space, utilizing either monocular or stereo cues.

1.3. Research Questions

The central research question of this thesis is: How can a robust and modular invisible-obstacle detection and distance estimation system be developed for low-altitude UAVs? This broad question is further broken down into the following sub-questions to guide the research:

1. What state-of-the-art deep learning architectures are most effective for the detection of thin objects in aerial imagery?
2. What are effective approaches for reliably estimating the distance of detected objects?
3. What method is the most effective for detecting and estimating the metric distance of the power lines in aerial images captured by UAVs?

1.4. Objectives and Scope

To effectively address the research question delineated in this dissertation, the study aims to develop a modular vision-based system for the detection and range estimation of power lines through the utilization of both monocular and stereo vision methodologies. The first objective involves developing an object detection and ranging pipeline, employing object detection-based deep learning models in combination with stereo vision for distance estimation. The other objective is to implement a semantic segmentation technique to detect the presence of power lines. The resultant segmentations are then subsequently integrated with the monocular depth estimation pipeline for the distance estimation. Lastly, the suitability and

effectiveness of both approaches for power line detection and range estimation are examined.

The scope of this dissertation is purposefully focused. It focuses on power line detection as a representative category of the slender, nearly invisible obstacles. This work does not address the comprehensive obstacle-avoidance control challenge or the broader category of other static or dynamic obstacles. Rather, the function of the developed system is exclusively perception (Detection and Distance estimation). The intended application is a low-altitude UAV equipped with a camera and inertial sensors. To summarize, this dissertation presents a modular perception system for detecting power lines and estimating their distance through visual processing and deep learning. By constraining the focus, the aim is to thoroughly investigate and produce results that are directly relevant to the safety of UAVs.

1.5. Thesis Structure

This thesis is structured into five chapters. Chapter 1 provides an introduction to the topic, outlining the background, problem statement, research questions, and objectives of the study. Chapter 2 reviews the existing literature on power line detection and distance estimation, while identifying current research gaps and positioning the current study within the broader context. Chapter 3 focuses on power line detection using object detection models in conjunction with stereo vision-based depth estimation, detailing the datasets utilized, model training, evaluation processes, and presenting results. Chapter 4 introduces a semantic segmentation approach for segmenting power lines, followed by a monocular 2D-to-3D reconstruction framework for their distance, along with training, evaluation, integration, and the results. Lastly, Chapter 5 concludes the thesis by presenting the key findings, reflecting on the overall contributions of the thesis, and suggesting directions for future work.

2. Literature Review

This chapter provides a comprehensive review of existing literature on power line detection and range estimation for autonomous aerial systems. It covers traditional approaches and current deep learning based methodologies for power-line detection and vision-based geometry for range estimation. The review highlights the methods and results achieved by various researchers and identifies gaps in current research. This review establishes the foundational context for the thesis's contribution, which involves developing deep learning-based models for power-line detection and employing monocular and stereo vision techniques for distance estimation.

2.1. Traditional Approaches

Classical computer vision methods for power-line detection relied on low-level image features and geometric transforms. Such algorithms often begin by detecting edges or linear segments by thresholding (Canny edge detection) and then linking points to form continuous wire segments (Hough transform) to find long lines.

Several scholars have built upon these foundations with specific improvements. For instance, M. Lehtomaki et al. [17] developed an automated powerline extraction algorithm employing a novel Connected Power-line search(CPLS) method and RANSAC line fitting, achieving 93.3% recall and 93.6% precision with 10cm tolerance across forested and open terrain. H. B. Kim et al. [18] formulated a classification framework that integrates point cloud data and object features through a random forest algorithm, which improved classification accuracy by 10% over single-classifier methods. Their framework proficiently identifies power lines based on distinct point attributes. In another direction, Z. Li et al. [19] developed a UAV-based power-line detection system by applying a pulse-coupled neural filter followed by an improved Hough transform. The proposed PCNF used the uniform brightness of power lines and their metal reflectance properties to

enhance edges, while knowledge-based clustering in Hough space reduced false positives from similar linear features. The method demonstrated robustness in clustered environments but required constant tuning for optimal PCNN output. In their research work, H. Sharma et al. [20] proposed a three-stage algorithm for power-line detection in UAV imagery. Their approach used adaptive thresholding with a morphological filter, achieving $<3\%$ false positives across 782 frames. The proposed method outperformed traditional Hough methods computationally while maintaining 60% line coverage per frame.

The traditional approaches work well where the line-background contrast is high and minimal visual complexity; however, they are highly sensitive to noise, variations in illumination, and the subtle visibility of slender wires. In their study, J. Song et al. [21] noticed that even though a Line Segment Detector(LSD) finds most power lines, it cannot easily distinguish true power lines from other linear background features(tower parts, house edges, etc). The Hough transform, although proficient in the detection of linear structures, incurs substantial computational expenses and may yield spurious detection in complex and cluttered backgrounds.

In practice, conventional methods (edge methods + Hough or LSD) often require the hand-tune adjustments of thresholds, or subsequent refinement, and they encounter limitations when obscured by vegetation, varying lighting, or sensor noise or sensor-induced noise influencing the visibility of cables.

2.2. Deep Learning Techniques

With the advent of deep learning in recent years, numerous works have sought to solve detection problems using deep learning techniques, The power-line detection is one such problem [22]. Provided enough training data. Learning-based methods now outperform classical computer vision methods. These methods typically use convolutional neural networks or transformers to directly learn wire features from images. Transfer learning is a popular technique, starting from a pretrained backbone and fine-tuning on power line data. To highlight the thin, curved structure of wires, recent deep learning work involves the use of specialized architectures and loss functions [23].

The study [24] introduced LS-Net, a single-shot line-segment detector trained on synthetic images. The network employs a four-grid approach to address discontinuities in detected lines. The proposed work outperformed traditional

approaches with a mean average precision of 0.82 and operates at 20.4 FPS. The work presented in [25] uses a pre-trained GoogLeNet model as a feature extractor and fine-tunes it for multi-class classification. It achieved an F-score of 75% and demonstrated applications like detecting missing power lines. J. Choi et al. [26] employed weakly supervised learning in conjunction with image-level labels and attention mechanisms for overhead power line detection, achieving a 2.74% improvement in F1-score over traditional methods. This approach remains effective in detecting thin obstacles using weak supervision. A. Stambler et al. [27] developed a DeepWire CNN to detect line segments in each image, followed by a 3D reconstruction pipeline to estimate the wire's vertical plane.

The research [28] framed power cable detection as a semantic segmentation task, leveraging dilated convolutional neural networks (CNNs) and synthetic data. To address the scarcity of annotated data, the authors generated a synthetic dataset of 67k images by rendering cables in 3D (catenary-shaped and straight lines) by ray tracing and overlaying them on real UAV data. The model achieved strong detection rates with 0.73 Average Precision (AP) and real-time inference at 3Hz on an NVIDIA Jetson TX2.

2.2.1. Detection-Based Methods

In 2024, T. Santos et al. [22] implemented YOLOv8 for power line detection in both visual and thermographic images, achieving high accuracy without manual feature extraction. The model was trained on visual and thermal UAV images and achieved robust performance with an mAP@0.5 over 90.5% on RGB images and even higher 96.9% on infrared images.

In a fusion-based approach, H. Aboalia et al. [29] developed three CNN models, including a VGG16-based architecture and a custom multi-input network designed for dual-sensor processing. Their trained VGG/AlexNet and a fused IR-RGB network achieved near-perfect classification accuracy (99%) on an IR-visible paired dataset. The novel multi-input network fuses high-resolution visible cues with high-contrast IR cues to generate an optimized detection map. Their fused model achieved 99.37% accuracy in power line detection, outperforming the single-stream networks.

In research work, [30], the authors presented an approach for aerial inspection of high-voltage power lines using YOLOv8. The proposed method involved fine-tuning the YOLOv8 model on the TIC (Towers, Insulators, and Conductors) dataset,

achieving an overall mean average precision (mAP@0.5) of 83.8%. Remarkably, the model ran at 243 fps on an RTX 3070 GPU, demonstrating YOLOv8's real-time capability, making it suitable for onboard UAV processing. Authors also showed that the learned YOLOv8 weights (trained for component detection) could be transferred to a defect-detection task (insulator cap loss), achieving a remarkable 99.5% mAP@0.5 and an F1-score of 99.89%.

Line-YOLO, introduced in [31], is an enhanced version of YOLOv8s-seg, specialized for power line angle detection. The model integrates deformable convolutions (DCNv4) to adapt to the variable curvatures and thickness of the sagging wires, BiFPN neck for efficient multi-scale fusion, and an EMA attention mechanism for improving target recognition in cluttered environments. For handling extremely small wire targets, a small target detection head was added. In the experiments on the custom data, Line-YOLO's mean average precision (mAP) increased by 6.2% over plain YOLOv8s-seg, with 28% fewer parameters and faster inference speed by 14 FPS compared to the baseline YOLOv8s-seg.

The research [32] introduced a YOLOv8-based advanced YOLO framework for UAV-based power line inspection. The authors enhanced the YOLOv8 model by adding directional filters to improve feature extraction for thin power lines, and output oriented bounding boxes (OBB) rather than axis-aligned boxes for more precise wire localization. The proposed framework was evaluated on the TTPLA dataset, achieving 78.24% AP@50 at 97 FPS, outperforming standard YOLOv8 and YOLACT++ in accuracy and speed.

2.2.2. Segmentation-Based Methods

Semantic segmentation utilizes advanced models such as Deep Convolutional Neural Networks (DCNNs) to achieve pixel-level segmentation of overhead power lines. These models can learn multi-scale features and perceive global information, rendering them highly effective in complex environments.

In their study, M. Cano-Solis et al. [33] proposed a methodology that employs DeepLabV3 and U-Net segmentation networks to differentiate power line corridors from vegetation backgrounds, achieving remarkable accuracy in pixel-level segmentation. M. Mao et al. [34] developed an end-to-end semantic segmentation model that integrates binocular vision alongside a feature pyramid network (FPN), facilitating precise and efficient identification of transmission lines. The study [35] proposed an advanced drone-based methodology for power line inspection

using image segmentation and adaptive visual control. YOLOv8 architecture was employed for the power line segmentation in urban scenarios. The study also introduced a custom dataset for power line segmentation. The research work [36] combined RGB and thermal imagery for enhancing the fault detection and segmentation accuracy of Greek power lines. The proposed methodology employed a hybrid approach that combined D-LinkNet for RGB image segmentation with Probabilistic Hough Transform for thermal data processing. This fusion improved line detection accuracy, achieving 0.99% and a precision of 0.80% in real-world tests.

The study [37] addressed the challenge of power line detection by proposing a deep-learning-based approach leveraging multi-modal sensor fusion. The Fast-SCNN architecture was modified by adding convolution layers and replacing bilinear interpolation with transposed convolution for finer up-sampling, while introducing a four-channel input that combines contrast-enhanced IR (using CLAHE) and VL for improving feature extraction. The methodology achieved a 6.7% higher mean intersection over union (mIoU) than the baseline, with real-time performance (24 fps), addressing class imbalance via a combined Jaccard and cross-entropy loss. Additionally, the paper also provided the first pixel-level matched VL-IR power line dataset, showing notable enhancements in detection accuracy and robustness under challenging conditions.

Some recent works also combine classical methods with deep networks. For example, W. Zhao et al. [38] integrated multi-scale line detection and semantic segmentation. The authors constructed a dataset comprising 409 UAV images classified by image clutter (IC) levels and supported a three-stage approach: (1) adaptive Gaussian pyramid-based LSD for candidate extraction, (2) an object-based MRF with a weighted region adjacency graph and GMM for spatial-spectral modeling, and (3) a simplified Kalman filter for segment linking. The method outperformed methods like IHT, CRT, and LSD-MRF, demonstrating superior performance by achieving up to 0.97 precision and 0.98 recall, especially in high-IC scenarios.

The paper [39] presented a framework for power line detection using UAV imagery, addressing the challenge posed by a noisy background. The proposed methodology integrated convolutional neural networks (CNNs) features with structured constraints, leveraging a multiscale hierarchical response derived from a modified VGG16 architecture. The technique fused the information from extracted semantic features with domain-specific knowledge, such as line continuity and

orientation, for enhancing detection accuracy and efficiency. When evaluated on two recently released datasets (PLDU and PLDM), it achieved strong performance in both precision and recall. On the PLDU dataset, it got an F1-measure of 0.914, surpassing contemporary state-of-the-art methodologies, hereby underscoring its relevance in autonomous inspection applications.

The study [40] introduced DUFormer, a semantic segmentation algorithm to detect power lines in aerial imagery. The algorithm combines a CNN-based feature extraction module with a transformer block for global modeling to address the challenge of identifying power lines against cluttered backgrounds. Experiments on the TTPLA dataset showed that it achieves a high accuracy of 84%. The other study [41] introduced PLGAN, a generative adversarial network (GAN)-based approach for power line segmentation. It used a GAN-based decoder to capture global context and thin-line geometry; on the TTPLA dataset, it yields very high precision 86% but moderate recall 58%. In [42], a specialized semantic segmentation model, PL-UNeXt, addresses the issues of intricate backgrounds and low pixel ratio of power cables. It focused on edge-detail segmentation, obtaining a 70.6 F1-score (+1.9%) on the TTPLA dataset and 68.41 mIoU (+5.2%) on the VITL dataset, without relying on infrared (IR) images.

2.3. Stereo vision based distance estimation

The paper [43] presented DROAN (Disparity-space Representation for Obstacle Avoidance) an innovative framework for detecting mapping, and avoiding slender obstacles like wires, which pose significant challenges for UAVs. The methodology combined monocular wire detection using a dilated convolutional neural network (CNN) with stereo-based generic obstacle perception, both of which were presented in disparity space for processing efficiency. The significant contribution was the C-space expansion technique, which took into consideration sensor uncertainty and robot size, for safe navigation. The proposed methodology employed the pose graph from disparity images for probabilistic occupancy inference along with a precomputed trajectory library for real-time collision-free planning. Experimental results demonstrate successful wire avoidance in real-time performance on an onboard TX-2 GPU, outperforming the detection capabilities of commercial systems like DJI Phantom 4, which fail to detect such thin obstacles.

The study [44] employed a trinocular system, which consisted of three calibrated cameras, for reconstructing power lines. The methodology involved the rectification

of all three images, followed by stereo matching for retrieving 3D curvature of the wire. By integrating GPS and feature-based localization techniques, the authors were able to calculate clearances to the ground or other potential obstacles. This methodology attained a high level of accuracy in safety monitoring, although it required the use of additional cameras.

The research in [45] presented a real-time stereo-based power line detection system using binocular vision and implemented on an FPGA. This study addressed the challenge associated with detecting thin power cables in low-texture environments through the stereo matching algorithm. Their methodology computed a dense disparity map from a stereo rig and subsequently incorporated a line-extraction step in the depth domain. In practice, they employed a robust stereo matching algorithm with cross-based support regions, followed by implementing a Hough transform on the 3D points to detect the presence of cable lines. This gave an estimate of the UAV-to-power line distance in real time.

The article [46] proposed an automated methodology for the inspection of power lines using UAV imagery, focusing on the Power Line Automatic Measurement Method based on Epipolar Constraints (PLAMEC) and the Semi Patch Matching technique, also based on Epipolar Constraints (SPMEC), for the purpose of three-dimensional reconstruction. The challenge of detecting thin power cables against a complex background was addressed by employing epipolar geometry along with gray ratio operators for extracting 2D power line vectors, which were subsequently transformed into 3D coordinates. Furthermore, the research also introduced a dense matching technique (SPMEC) aimed at reconstructing ground point clouds for obstacle detection, achieving notable precision with a root mean square error in height of 0.233 m for the point cloud and 0.205 m for the power lines. Evaluations demonstrated the effectiveness of the methodology, yielding a 93.2% success rate in the detection of power lines and an obstacle detection accuracy maintained within ± 0.5 m in comparison to field measurements.

2.4. 2D-to-3D Reconstruction of Linear Structures

The study [47] presents a real-time, model-based methodology for 3D wire reconstruction using a monocular camera and catenary curve model. The authors exploit the fact that power lines sag in a catenary shape under gravity and assume a sequence of camera frames with known poses. The proposed technique leverages pixel-wire powerline segmentations from a CNN and formulates the

reconstruction process as a nonlinear least squares optimization problem, and minimizes a distance-transform reprojection error. The key innovation within this framework was the implementation of a five-parameter catenary model, which gave a continuous 3D wire estimate, robustly bridging gaps from occlusions or missing detections. This study advanced the thin-obstacle perception in UAVs by integrating deep learning with geometric priors. Through only optimizing 5 parameters instead of thousands of free points, it offered a lightweight solution for power line inspections.

The authors in their research [48] addressed the challenge of detecting thin obstacles in autonomous mobile robots. They proposed a vision-based approach utilizing edge detection and edge-based visual odometry techniques with or without stereo. Their stereo solution utilized disparity maps to get the scale information, while the monocular solution incorporated IMU data to recover the scale. Experimental findings show superior performance over SGBM and LSD-SLAM baselines. The system also showed that the edge-based VO can recover the cable's 3D curves in real time with a 17fps on a Celeron PC.

The work [49] presented a method for the power line inspection with a particular focus on 3D reconstruction and corridor clearance monitoring. The presented methodology incorporated decorrelation stretching, an enhanced Prewitt filter, and Random Sample Consensus (RANSAC) for the detection of power lines, subsequently employed epipolar geometry for 3D reconstruction. The achieved results were compared with terrestrial scanning (TS) and terrestrial laser scanning (TLS) data and demonstrated a comparable accuracy. The authors reported reconstructions within a few centimeters of ground truth with an RMSE 2–6 cm over spans of hundreds of meters.

The article [50] examined the complexities associated with power line 3D reconstruction using Lidar-equipped drones. The authors proposed a novel approach based on domain-specific hypotheses such as images are acquired orthogonally to the power lines, and the wires are situated within vertical planes. This approach integrated the structure from motion for initial camera pose estimation with a minimization problem aimed at refining poses and reconstructing wires by projecting segmented 2D curves onto 3D planes. The authors reported a 3D wire accuracy on the order of 10m (for a 250m span, error 7–12m).

P. Wang et al. [51] conducted a study on the monocular reconstruction of slender objects (tree branches, wires) leveraging curve-based structure-from-motion. Through the iterative optimization of camera poses and 3D curve networks, they

effectively minimized reprojection errors of skeletal curves across various camera frames. Their proposed approach accurately addressed issues of self-occlusion and topology preservation, thereby achieving high-fidelity reconstructions of textureless slender objects without dependence on depth data or textured backgrounds.

The article [52] introduced DLT-combined-lines, a novel technique for estimating camera pose, derived from line correspondences (PnL) by merging DLT-Lines (which utilized 3D points) and DLT-Plücker-Lines (which employed Plücker coordinates). This hybrid methodology diminishes the minimum requisite lines to five while enhancing accuracy by capitalizing on redundant 3D structures and multiple pose estimates derived from a combined projection matrix. This methodology achieved state-of-the-art performance in positional estimation and projection error when dealing with extensive line sets, particularly in the presence of substantial noise, while sustaining high computational efficiency (12 milliseconds for 1000 lines).

Another study [53] introduced a UAV-based oblique photogrammetry system aimed at the 3D reconstruction of transmission lines, integrating a dual-camera configuration and a dual-strip flight mode for acquiring multi-directional aerial imagery. This proposed approach mitigated the need for ground control points (GCPs) by employing differential Global Navigation Satellite System (GNSS), thereby attaining sub-meter precision in geo-referencing. This research underscored the efficacy of oblique imaging for off-site inspections; however, the manual digitization of power pylons continues to present a significant challenge. Conversely, [54] presented line-based Structure from Motion pertinent to urban scenes, leveraging orthogonal vanishing directions to restrict 3D line parameters during the optimization process. This methodology automated the detection and classification of lines through an expectation-maximization (EM)-based vanishing point estimation technique, thereby reducing the degrees of freedom in the reconstruction process. Through constraining gauge freedoms (e.g., vertical/horizontal lines), the convergence and precision in sparse 3D models were enhanced.

2.5. Research Gaps

Despite significant advancements in UAV-based power-line inspection, several critical gaps remain in methods focused specifically on power-line detection and distance estimation. Our survey highlights four main areas where existing work falls short:

1. **Inspection vs. Detection–Ranging Focus**

Most studies emphasize fault inspection, identifying corrosion, wear or anomalies, rather than accurate detection of power lines coupled with distance estimation. While deep-learning models can segment or classify power-line pixels, they are rarely evaluated for precise localization in image space, nor integrated with depth information to achieve metric distance measurements.

2. **Distance Estimation and 3D Localization**

Current 3D reconstruction efforts rely on offline stereo or LiDAR scans to map line geometry post-flight. Such methodologies are unsuitable for onboard, frame-by-frame distance measurement. Classical stereo and 3D reconstruction methods have not been adapted to accommodate the minimally textured, slender profile characteristic of power lines, and monocular depth models trained on generalized scenes do not extend effectively to slender cables. To the best of our knowledge, no established method reliably produces per-pixel or per-segment depth estimations for wires utilizing a single camera.

3. **Data Scarcity and Viewpoint Limitations**

Existing datasets predominantly center on close-range, top-down views for defect detection. They lack front-facing, long-range sequences that include annotated line segments alongside corresponding ground-truth distance data. It is difficult to train and evaluate detection and ranging networks without a benchmark that pairs power-line bounding boxes or masks with depth labels across varied altitudes, lighting, and backgrounds.

4. **End-to-End Integration**

While individual modules such as deep learning detectors, stereo matchers, and monocular depth CNNs exist. The research lacks the development of a cohesive pipeline that processes raw video, detects power lines, estimates their distances in real-time, and delivers actionable range data. Lightweight implementations suitable for UAV hardware, acceptable latency and accuracy for operational flight, remain undeveloped.

3. Power line Detection and Distance Estimation using Object Detection and Stereo Vision

3.1. Datasets for Object Detection

The proposed research employed two distinct datasets for the training and evaluation of the deep learning models:

1. **PLdataset:** The publicly accessible PLdataset which was introduced in [35] study. This dataset comprises a collection of images derived from the TTPLA [14] and PLD-UAV datasets [55], along with the data captured through a Parrot Bebop-2 drone, featuring a total of 3,319 annotated images acquired from low-altitude aerial vehicles. However, this dataset was limited in scope as it contains only top-down views, mostly used for power line inspection methodologies. For the scope of our work, forward-looking aerial views are required.
2. **Custom Dataset:** To get the front-faced images, a new dataset was captured in the Smart Mechatronics and Robotics lab utilizing a Stereolabs ZED2 stereo camera. Synchronized stereo video(30 FPS, 1920×1080) of power cables was recorded in the lab from multiple frontal viewpoints. The ZED2's wide baseline (120 mm) and known calibration allowed later metric measurements. This custom dataset encompasses full-frame imagery that includes power lines positioned at various distances and angles, ensuring a diverse training and evaluation sample range.

3.1.1. Annotations

The labeling process was performed in the Roboflow online application, which can augment data in the images to improve the model's generalization and achieve greater performance on unseen images. In certain images, it can be challenging to correctly annotate the power lines due to the background. All annotations were exported in YOLO format to ensure compatibility with established object detection frameworks. A total of 1,800 images were annotated. The resultant dataset was methodically partitioned into training (70%), validation (20%), and test (10%) subsets, thereby preserving a balanced distribution of powerline configurations across all divisions.

3.2. Object Detection Models

The next stage after data collection and data labeling is to choose suitable deep learning model for the power line detection. Due to the limited size of the dataset, transfer learning can be leveraged by using state-of-the-art object detectors pre-trained on large datasets such as ImageNet and COCO (Common Objects in Context) [56]. The object detectors are classified into two main types: one-stage and two-stage detectors, as shown in Figure 3.1. Two-stage detection frameworks like Faster R-CNN, initially produce a collection of region proposals, subsequently classifying and refining these proposals in a second phase. Conversely, one-stage detection frameworks, such as YOLO and SSD, directly predict class probabilities and bounding box coordinates from the input image, thereby eliminating the necessity for a discrete region proposal stage.

The focus herein is placed on YOLO object detectors, attributable to their rapidity, efficiency, and efficacy in the context of this particular task. At its core, the YOLO architecture is constituted by three fundamental components. The backbone, which is a convolutional neural network, is responsible for encoding image information into feature maps at varying scales. The neck then processes these feature maps, a series of layers designed to integrate and refine feature representations. Ultimately, the head module produces predictions pertaining to object bounding boxes and class labels based on the processed features [1]. In this study, YOLOv5, YOLOv8, and YOLOv11 have been selected owing to their real-time operational capabilities and improved accuracy relative to their predecessors and successors. The selection of these models was based on their

widespread usage, performance metrics, and potential appropriateness for the task of power line detection.

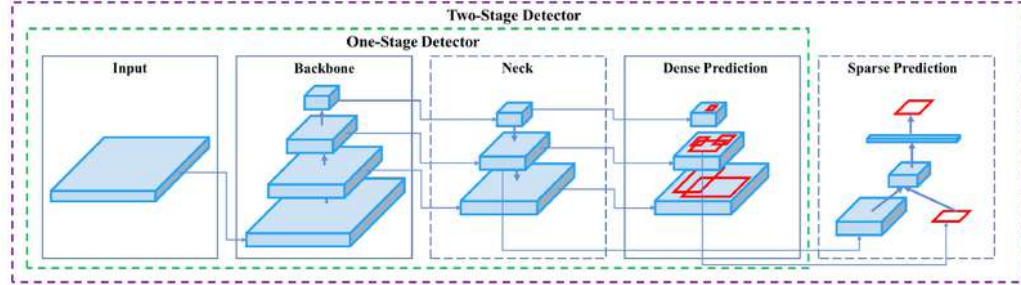


Figure 3.1: Object Detection Models [1]

3.2.1. You Only Look Once (YOLO) version 5

YOLOv5 was released in 2020 by Glen Jocher, founder and CEO of Ultralytics [57], transitions from Darknet to PyTorch. The architecture consists of the backbone, the neck, and the head. The backbone is based on CSPDarknet53 [58], which employs Conv-BN-SiLU (CBS) modules and C3 bottleneck blocks [59]. The C3 blocks use Cross-Stage Partial (CSP) connections, splitting feature maps into two paths, one processed through a series of convolutions and the other left unchanged, before merging them. This design improves gradient flow and reduces computational overhead while maintaining feature richness. All convolutions use the SiLU activation function for smoother gradient propagation.

The neck incorporates a Feature Pyramid Network (FPN) and Path Aggregation Network (PANet) [58], enhancing multi-scale feature fusion by combining top-down and bottom-up pathways. Additionally, the Spatial Pyramid Pooling - Fast (SPPF) module replaces the older SPP layer. It aims to speed up the computation of the network without sacrificing receptive field coverage by pooling features of different scales into a fixed-size feature map [57].

The detection head replicates the multi-scale YOLOv3-style detection head (anchor-based) for bounding-box regression and classification. YOLOv5 is anchor-based (like YOLOv3), with combined output layers for objectness, class scores, and box offsets. It uses a combination of CIoU loss for bounding-box regression and binary cross-entropy for classification (as default) [57]. The final output undergoes Non-Maximum Suppression (NMS) to eliminate redundant detections.

3.2.2. You Only Look Once (YOLO) version 8

YOLOv8, released in January 2023 by Ultralytics, marks a significant progression within the YOLO series. It is renowned for its balance between detection accuracy and speed. YOLOv8 architecture Figure 3.2 is based on YOLOv5 design but with several key changes.

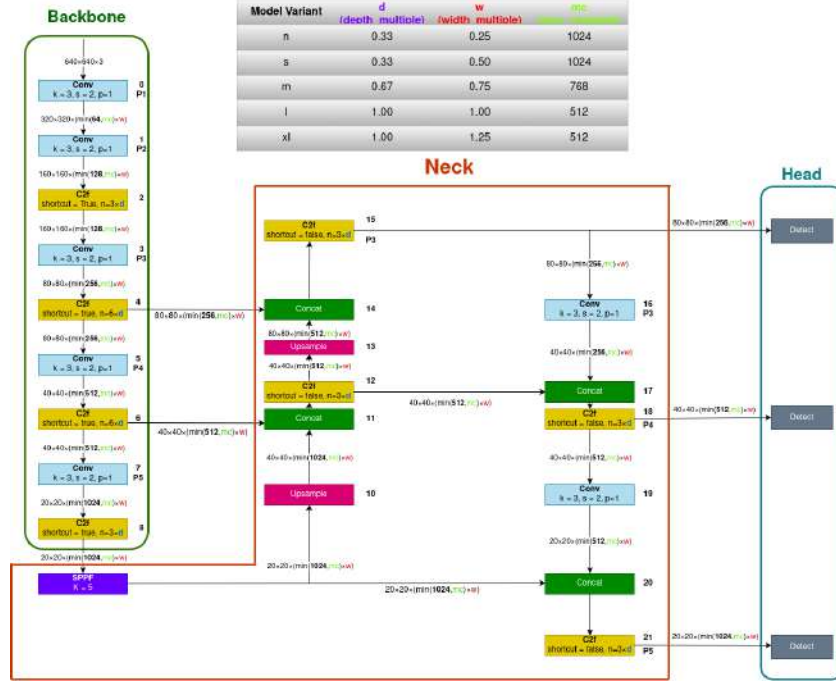


Figure 3.2: YOLOv8 Architecture

Backbone

The backbone serves as the fundamental structure of the YOLOv8 model, responsible for the extraction of salient features from the input image. Just like the YOLOv5, it is also based on the CSPDarknet53 architecture. The YOLOv8 backbone includes Conv Block. These layers are designed to discern local patterns, including edges, textures, and shapes, by the application of filters to the input image. It comprises a 2d convolution, batch normalization, and a sigmoid linear unit activation (SiLU) function. The mathematical expression for the convolution operation is as follows:

$$(f * g)(i, j) = \sum_m \sum_n f(m, n) \cdot g(i - m, j - n) \quad (3.1)$$

where f is the input image, g is the filter, and (i, j) denotes the pixel position.

The Sigmoid Linear Unit (SiLU) activation function is defined as:

$$\text{SiLU}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \quad (3.2)$$

where x is the input to the activation function, and $\sigma(x)$ is the sigmoid function.

Furthermore, The Backbone also contains C2F Block. The Cross Stage Partial Fusion (C2f) block represents an extension of the CSPNet paradigm. The backbone utilize C2f for feature extraction at all stages. The C2f block consists of two ConvModules and a variable quantity of Bottleneck blocks, interconnected through Split and Concat operations. The Bottleneck block functions as a type of residual block utilized within the C2f block. It helps reduce the dimensionality of the feature maps while maintaining critical information.

Neck

The neck of the YOLOv8 architecture connects the backbone (which extracts features from the image) to the head (which makes predictions like object classes and bounding boxes). The neck fuses the multi-scale features generated by the backbone before passing them to the detection head module. The YOLOv8 backbone consists of Spatial Pyramid Pooling Fast (SPPF), which represents a modification of SPP or Spatial Pyramid Pooling, available in the predecessors of YOLOv8. SPPF constitutes a block that permits feature maps to be represented across multiple scales. By pooling at varying scales, SPPF allows the model to capture features at different levels of abstraction. The Neck also consists of a Path Aggregation Network (PAN), which uses a PANet structure to fuse multi-scale features. Through the combination of both high-level semantic information, which captures the overall context of the scene, and low-level spatial information, which preserves finer details, the PANet enhances YOLOv8's ability to detect objects at various scales.

Head

The head of the YOLOv8 is anchor-free and decoupled. Unlike YOLOv5's single-head, YOLOv8's detection block consists of two components, specifically the classification head and the regression head. It generates the network's outputs, including bounding boxes, object classes, and confidence scores. The head consists of a Bounding box prediction which predicts bounding boxes directly using the following equations:

$$t_x = x, \quad t_y = y, \quad t_w = \exp(w), \quad t_h = \exp(h) \quad (3.3)$$

where t_x , t_y represent the predicted center coordinates, and t_w , t_h are the width and height of the predicted bounding box.

The head also gives the confidence score, which reflects the probability of an object being present within a predicted box, and is computed using a sigmoid activation function:

$$\text{Confidence} = \sigma(C) \quad (3.4)$$

where C is the raw confidence output.

Finally for the class prediction, YOLOv8 applies a softmax function over the set of class scores to estimate the probability of each class:

$$P(\text{class } i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (3.5)$$

where x_i represents the raw score for class i , and $P(\text{class}_i)$ is the predicted probability that the detected object belongs to that i class.

Loss Functions in YOLOv8

1. Classification Loss

- **Binary Cross-Entropy (BCE) Loss:** For classification, the model applies BCE Loss, which measures the discrepancy between the predicted class probabilities and the ground truth labels as follows:

$$L_{\text{cls}} = - \sum_{c=1}^C [y_c \log(p_c) + (1 - y_c) \log(1 - p_c)] \quad (3.6)$$

where y_c represents the ground truth label and p_c represents the predicted probability for class c .

2. Bounding Box Regression Loss

- **Complete IoU (CIoU) Loss:** YOLOv8 uses the Complete Intersection over Union (CIoU) Loss to regress bounding box coordinates, which takes into account aspect ratio consistency, the distance between box centers, and the overlap area:

$$L_{\text{CIoU}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{\text{gt}})}{c^2} + \alpha v \quad (3.7)$$

Here, ρ denotes the Euclidean distance between the centers of the predicted box b and the ground truth box $\mathbf{b}^{\text{gt}}$, c is the diagonal length of the smallest enclosing box covering both, α is a balancing parameter, and v measures the aspect ratio consistency.

- **Distribution Focal Loss (DFL):** To improve bounding box regression, YOLOv8 uses a Distribution Focal Loss (DFL) by reducing the distributional error in predicted box positions:

$$L_{\text{DFL}} = \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^B [p_{ij} \log(p_{ij}) + (1 - p_{ij}) \log(1 - p_{ij})] \quad (3.8)$$

where N is the number of bounding boxes, B is the number of bins, and p_{ij} is the predicted probability of the i_{th} bounding box falling into the j_{th} .

The final loss function guiding the training process is a weighted sum of these individual losses:

$$L = \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{CIoU}} L_{\text{CIoU}} + \lambda_{\text{DFL}} L_{\text{DFL}} \quad (3.9)$$

where λ_{cls} , λ_{CIoU} , and λ_{DFL} are the respective weights controlling the contribution of each component.

3.2.3. You Only Look Once (YOLO) version 11

YOLOv11, released in September 2024 by Ultralytics, is the latest advancement in the YOLO series of real-time object detection models. Building on the achievements of earlier versions, YOLOv11 incorporates several novel architectural improvements that significantly enhance detection accuracy.

The architecture of YOLOv11 refines YOLOv8's design. Its backbone is also derived from CSPDarknet53, but with new CSP variants. A significant advancement is introducing the C3k2 block (Cross-Stage Partial with a kernel size of 2), which replaces the older C2f blocks. The C3k2 block employs two smaller convolutional operations in place of a singular large operation, thereby reducing computational demands while maintaining representational power [1]. YOLOv11 continues to utilize the rapid Spatial Pyramid Pooling (SPPF) layer and introduces a C2PSA (Cross-Stage Partial with Spatial Attention) module after it. The C2PSA block implements spatial attention mechanisms, directing the network's focus toward critical regions—a feature that is notably absent in YOLOv8 [1].

The neck similarly incorporates upsampling and concatenation; however, all CSP blocks (previously designated as C2f) are substituted with the innovative C3k2 blocks for enhanced efficiency [1]. The head consists of multiple detection branches; each branch employs C3k2 blocks to process feature maps at varying depths. Following these operations, YOLOv11 integrates several CBS layers (Convolution + BatchNorm + SiLU) to further optimize features [1]. These final layers serve to stabilize feature scaling and introduce non-linearity through the SiLU activation function. The detection head subsequently utilizes Conv2D layers to predict bounding box offsets, objectness scores, and class probabilities [1]. Consistent with YOLOv8, the outputs for objectness and classification leverage sigmoid/softmax activations. YOLOv11 maintains an anchor-free paradigm, employing a decoupled head architecture.

3.3. Model Training

We trained all three models, YOLOv5, YOLOv8, and YOLOv11, using the same pipeline to facilitate equitable comparison and ensure reproducibility. The model's training process was carried out on Google Colab using a T4 GPU alongside an Intel Core i5 CPU. All models were initialized using pre-trained weights on the COCO dataset, followed by fine-tuning on our power line dataset. The hyperparameters used for training all models are given in Table 3.1. These hyperparameters were meticulously chosen based on the recommendations from the respective official Ultralytics repositories.

Table 3.1: Training hyperparameters for YOLOv5, YOLOv8, and YOLOv11 models.

Hyperparameter	YOLOv5	YOLOv8	YOLOv11
Image Size (px)	640×640	640×640	640×640
Optimizer	SGD	AdamW	AdamW
Learning Rate	0.01	0.002	0.002
Batch Size	16	16	16
Number of Epochs	50	50	50
Decay	0.0005	0.0005	0.0005
momentum	0.937	0.9	0.9

3.4. Object Detection Performance Metrics

For evaluating the performance of the models for detecting the power lines, we utilized a set of standard evaluation metrics. Table 3.2 summarizes these metrics along with their descriptions and formulas.

3.5. Distance Estimation

The next stage after the power line detection is distance estimation. Among the 3 trained YOLO models, we selected YOLOv8 for deployment in the stereo vision-based distance estimation pipeline due to its superior balance between inference speed and detection accuracy. As depicted in Figure 3.3, the YOLOv8 model processes the input frames from the ZED2 camera to detect power lines, producing predicted bounding boxes around the detected wires. These detections are subsequently used in a stereo vision system for estimating the distance of the power cables.

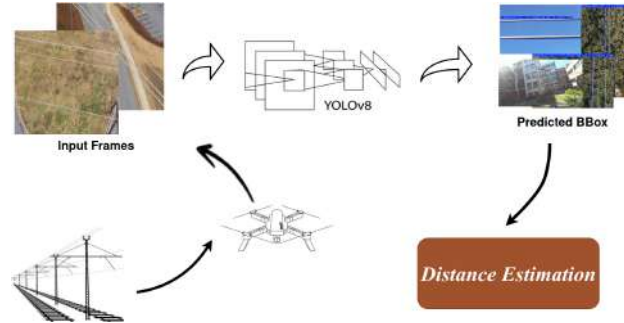


Figure 3.3: Schematic Representation of Power Line Distance Estimation

3.5.1. Stereo Camera Setup

The Stereolabs ZED 2 stereo camera was employed to implement the stereo vision-based object detection and distance estimation system. The ZED 2 represents a cutting-edge passive stereo camera, featuring dual RGB sensors that enable the acquisition of high-resolution stereo imagery through synchronized frame capture, thereby rendering it particularly adept for depth perception tasks. The comprehensive software development kit (SDK) and inherent support for the Robot Operating System (ROS) further enhance its capacity for seamless incorporation into robotic frameworks. The factory-calibrated baseline (center-to-center distance

Table 3.2: Evaluation metrics used for assessing the object detection models.

Metric	Description	Formula
Precision	Measures the proportion of correctly predicted positive detections out of all positive predictions.	$\frac{TP}{TP+FP}$
Recall	Measures the proportion of actual positive instances that were correctly predicted.	$\frac{TP}{TP+FN}$
Accuracy	Represents the proportion of total correct predictions (both true positives and true negatives) out of all predictions made.	$\frac{TP+TN}{TP+TN+FP+FN}$
F1 Score	Harmonic mean of precision and recall. It balances both false positives and false negatives.	$2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
mAP@0.5	Mean Average Precision computed at an Intersection over Union (IoU) threshold of 0.5.	$\text{mAP}_{0.5} = \frac{1}{N} \sum_{i=1}^N AP_{i,0.50}$
mAP@0.5:0.95	Mean Average Precision averaged over multiple IoU thresholds from 0.5 to 0.95 with a step size of 0.05.	$\text{mAP}_{0.5:0.95} = \frac{1}{N} \sum_{i=1}^N \left(\frac{1}{10} \sum_{j=0.50}^{0.95} AP_{i,j} \right)$

Where: TP = True Positives, FP = False Positives, TN = True Negatives, FN = False Negatives, AP = Average Precision, N = Number of classes or detections, and IoU = Intersection over Union.

between the left and right cameras), measures 120 mm, with configurable options available for both image resolution and frame rates [60]. In our work, stereo images were acquired at a resolution of 1920×1080 pixels, with a frame rate of 30 fps. Data collection was conducted through the recording of stereo video in ROS2, formatted to ROSBAG, thereby ensuring synchronized rectified image pairs for each corresponding timestamp. The intrinsic and extrinsic calibration processes are managed by the ZED SDK, which facilitates the provision of the stereo imagery necessary for later depth estimation.

3.5.2. Stereo-Vision and Depth Estimation

Stereo vision determines 3D depth through triangulation of matching points in the left and right camera images. The methodology of stereo vision is inspired by the biological process and employs two horizontally aligned cameras for perceiving the depth of the object by comparing the location of corresponding features within the left and right images. This paradigm mimics the biological mechanism of stereopsis, wherein the human visual system derives depth information by analyzing the subtle variations in perspective between the observations of each eye. The foundational principle relies upon the geometric relationship established between two cameras that are positioned horizontally apart from one another, thereby creating a baseline distance that permits triangulation of 3D points in space. When a point $P(X, Y, Z)$ in real world is perceived by both cameras, it is projected onto corresponding image coordinates u_L, v_L in the left camera and u_R, v_R in the right camera, as depicted in Figure 3.4.

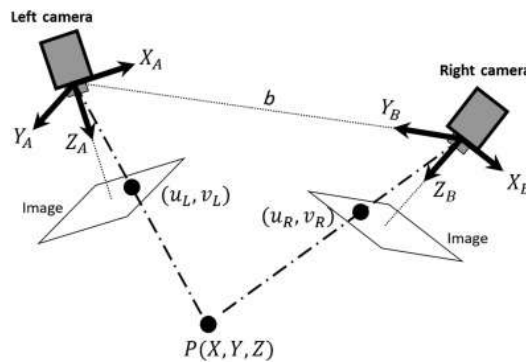


Figure 3.4: Typical Stereo-vision System

The core principle entails the identification of corresponding features within the left and right images and the computation of their disparity. This disparity is

directly related to the depth of the point observed within the scene. In order to simplify the correspondence search, stereo images are often rectified. Rectification is a pre-processing step that geometrically modifies the images so that their epipolar lines are aligned and rendered horizontal. These transformations ensure that corresponding points reside on the same scanline across both images, thereby confining the search for matches one-dimensional problem along horizontal lines. This rectification procedure capitalizes on the epipolar geometry inherent to stereo imaging, which restricts the locations of corresponding points to epipolar lines due to the coplanarity of the camera centers and the 3D point being observed.

Within the rectified stereo images, the disparity d between corresponding points is computed as the difference in their horizontal coordinate because vertical disparities are eliminated :

$$d = x_L - x_R \quad (3.10)$$

where x_L and x_R denote the horizontal pixel coordinates of the corresponding point in the left and right images, respectively.

Subsequently, the depth Z of the point within the scene can be computed through the triangulation principle:

$$Z = \frac{f \cdot b}{d} \quad (3.11)$$

Here, f shows the focal length of the cameras, b represents the baseline distance between the centers of the two cameras, and d denotes the computed disparity.

This equation indicates that depth is inversely related to disparity: as disparity increases (indicating that the point is nearer to the cameras), depth decreases, and vice versa. The accuracy of depth estimation is thus dependent on the precise calibration of the stereo camera system to determine f and b , as well as robust algorithms for disparity computation.

3.5.3. Power Line Distance Estimation Integration Pipeline

The comprehensive integration pipeline developed for power line detection and distance estimation is illustrated in Figure 3.5. It fuses YOLOv8 with stereo vision for the distance estimation of power lines. The methodology begins with capturing synchronized and rectified left and right camera image pairs. Afterwards, YOLOv8 detection module is employed on the left image to identify and localize power lines. The output encompasses bounding boxes for detected power lines. For each detection, a corresponding region of interest (ROI) is cropped in both left

and right images. This mapping is enabled by the established stereo rectification, which permits the accurate mapping of bounding boxes from the left image to the right image.

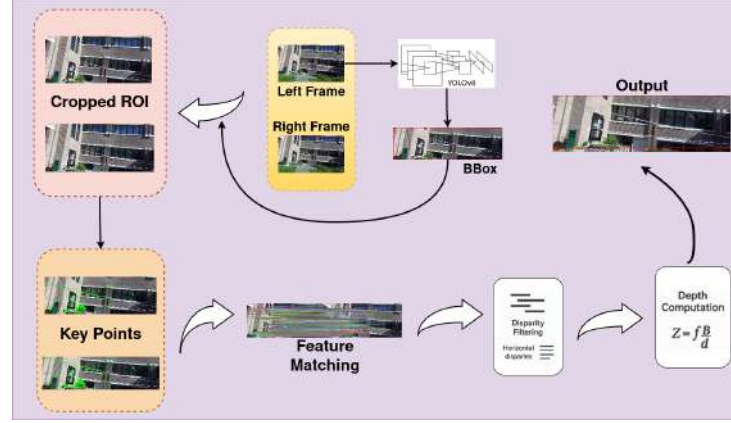


Figure 3.5: Stereo vision-based distance estimation methodology

Within each defined ROI, we extract features using ORB (Oriented FAST and Rotated BRIEF) feature matching. The ORB feature matching is selected due to its efficiency and processing speed [61], making it suitable for real-time applications. Once features are detected, keypoints and their descriptors are computed within the ROIs. Features are subsequently matched within both ROIs utilizing the brute-force Hamming distance matcher. These pairs of keypoints are then employed for computing disparity, defined as the difference in horizontal coordinates between matched keypoints in the left and right images. Given that the stereo images utilized are rectified, vertical disparity remains negligible, thereby simplifying the computational process. The depth of the power lines is then estimated through utilizing the stereo pinhole model, which correlates disparity to depth via Equation 3.11. Depth is computed for all valid keypoint correspondences within the ROI, and the median of these values is calculated as the definitive distance estimate. This median filtering technique mitigates the influence of outliers and yields a more stable and credible depth measurement. The estimated distance is subsequently allocated to the corresponding detection and visualized through the rendering of the bounding box on the left image, accompanied by an annotation indicating its computed distance in meters. This integration is formalized in 1.

Algorithm 1 Power Line Distance Estimation Integration Pipeline

```

1: procedure STEREODISTANCEESTIMATION
2:   Load YOLO model
3:   Set baseline, focal_length
4:   left, right  $\leftarrow$  LoadStereoImages()
5:   bboxes  $\leftarrow$  YOLODetect(left)
6:   results  $\leftarrow$  [ ]
7:   for each bbox in bboxes do
8:     roi_left  $\leftarrow$  Crop(left, bbox)
9:     roi_right  $\leftarrow$  Crop(right, bbox)
10:    (kp1, des1)  $\leftarrow$  ORB(roi_left)
11:    (kp2, des2)  $\leftarrow$  ORB(roi_right)
12:    if descriptors exist then
13:      matches  $\leftarrow$  MatchFeatures(des1, des2)
14:      disparities  $\leftarrow$  [kp1[i].x - kp2[i].x for i  $\in$  matches]
15:      disparity  $\leftarrow$  Median(disparities)
16:      depth  $\leftarrow$   $\frac{focal \times baseline}{disparity}$ 
17:      Append (bbox, depth) to results
18:    end if
19:  end for
20:  VisualizeResults(results)
21: end procedure

```

3.6. Results

This section delineates the results of the power line detection and distance estimation framework, which combines the YOLO-based object detection with stereo vision-based depth estimation.

In order to determine the most effective model for power lines, three variations of the YOLO architecture, YOLOv5, YOLOv8 and YOLOv11, systematically trained and tested on the same dataset. The performance metrics of each model on the test subset are illustrated in Table 3.3 and Figure 3.6, utilizing the object detection metrics delineated in section 3.4. YOLOv8 showed the most superior overall performance, achieving the highest precision (0.99) and mAP@0.5 (0.99), thus signifying robust detection reliability. Additionally, it provides an optimized

inference time suitable for real-time applications. In light of this performance, YOLOv8 was designated as the detection backbone for the stereo-based distance estimation framework.

Table 3.3: Comparison of Model Performance Metrics

Metric	YOLOv5	YOLOv8	YOLOv11
Precision	0.98	0.99	0.95
Recall	0.97	0.96	0.95
mAP@0.5	0.98	0.99	0.97
mAP@0.5:0.95	0.88	0.91	0.86
F1 Score	0.97	0.97	0.96

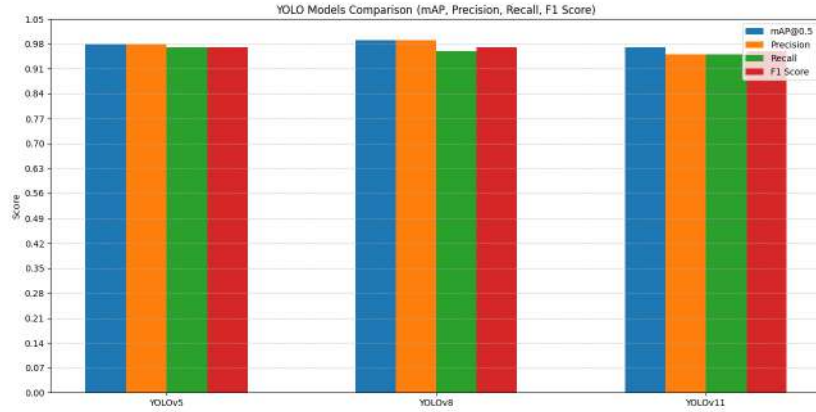


Figure 3.6: Comparison of YOLO Models

Upon successful detection of power lines in each pair of stereo images through YOLOv8, stereo correspondence techniques were used to determine the distance to the identified power lines. To assess the accuracy of the estimated distances, data was obtained by measuring actual distances, which included ground truth distances of 2 meters, 3.5 meters, and 7 meters from the camera to the power lines. The distance estimation results corresponding to the ground truth values of 2 meters, 3.5 meters, and 7 meters are depicted in Figure 3.7, Figure 3.8, and Figure 3.9, respectively. In these figures, the red bounding boxes represent the detected power lines, and the text indicates the estimated distance.

The results show that the proposed system can detect and localize power lines with a high degree of accuracy in real-world scenarios. The maximum absolute error at shorter distances was around 0.11 meters, which is deemed acceptable for obstacle avoidance tasks within low-altitude UAV navigation. It should be noted that the error exhibits a slight increase at greater distances, attributable

to diminished disparity resolution, a known limitation inherent to stereo vision systems. However, even at a distance of 7 meters, the maximum absolute error was around 0.15 meters, indicating that the system retains its reliability for the location of the medium-range power line.



Figure 3.7: Detected power lines with the estimated distance (GT = 2 m)



Figure 3.8: Detected power lines with the estimated distance (GT = 3.5 m)



Figure 3.9: Detected power lines with the estimated distance (GT = 7 m)

In our stereo-based distance estimation methodology, the bounding boxes produced by YOLOv8 define the Region of Interest (ROI) for further computations.

The precision of depth estimation is considerably affected by the dimensions and positioning of these bounding boxes. In scenarios where power lines are securely suspended and horizontally aligned, YOLOv8 typically generates tightly-fitting bounding boxes that accurately encompass solely the cable structure (Refer to the 3.7, 3.8, 3.9). This results in the disparity computation being confined to the pixels representing the cables, thereby producing reliable estimates of depth.

Conversely, in instances where cables exhibit sagging, are loosely aligned, or experience occlusion, the detector produces bounding boxes that are considerably larger than the actual region occupied by the power line Figure 3.10. These boxes may encompass substantial portions of the background (e.g., building walls, sky), given that power lines generally occupy only a minimal number of pixels in the image. Consequently, the generated disparity map becomes tainted by background features, resulting in estimated distances that are less precise or entirely erroneous.

This complication arises from the fact that the disparity map is computed exclusively within the ROI defined by the bounding box. If the bounding box incorporates high-contrast background regions rather than the cables, the disparity algorithm is inclined to prioritize those dominant features, leading to distorted depth values. These observations underscore the paramount significance of bounding box accuracy when addressing thin wires, wherein a minor error in bounding box dimensions can precipitate disproportionately substantial inaccuracies in depth estimation. The supplementary results showing power line detection and ranging in real-world scenarios are available at: Supplementary Results.



Figure 3.10: Impact of bounding box variations on distance estimation accuracy

4. Semantic Segmentation and 2D-3D Reconstruction of Power Lines

4.1. Segmentation Datasets

The datasets employed for the segmentation task were congruent with those employed in the object detection phase section 3.1, but re-annotated for pixel-wise segmentation. This approach ensured consistency in data sources while adapting the annotations to fulfill the specific demands of semantic segmentation models.

1. **TTPLA:** We employed the publicly accessible TTPLA dataset (Transmission Tower and Power Line Aerial images), introduced in [14], which comprises of 1,100 high-resolution aerial images ($3,840 \times 2,160$ pixels) with 8,987 manually annotated instances of transmission towers and power lines, all provided in COCO format. The images were extracted from 80 UAV-recorded videos captured across diverse geographic locations and viewing angles, making the dataset well-suited for both instance segmentation and object detection tasks.
2. **Custom Dataset:** The same custom dataset acquired at SMART lab section 3.1 was also used for the segmentation task.

4.1.1. Annotations

Manual polygonal annotations were created using the Roboflow platform for the segmentation task. In contrast to the bounding box annotations employed for object detection, segmentation annotations involved drawing precise outlines around power lines to create accurate masks. These masks were subsequently exported in COCO segmentation format to ensure compatibility with prevalent segmentation architectures.

The combined dataset encompasses 1,800 annotated images, mirroring the object detection dataset in terms of image content yet differing in annotation format. The dataset was once again split into training (70%), validation (20%), and test (10%) subsets, ensuring a balanced representation of power line shapes and backgrounds for training robust segmentation models.

4.2. Deep Learning based Segmentation Models

Recent advancements in deep learning methodologies have facilitated the accurate detection of slender linear entities, such as power lines, within aerial imagery. Convolutional Neural Networks (CNNs) have been extensively utilized for this specific task owing to their ability in learning hierarchical features directly from the provided data. The introduction of Fully Convolutional Networks (FCNs) constituted a significant milestone by permitting end-to-end pixel-wise segmentation, thereby converting conventional classification networks into models designed for dense prediction.

In the context of powerline detection, pixel-level segmentation is of paramount significance, as power lines typically appear as very thin structures that span across the image with only a few pixels. Object detection methods reliant on bounding boxes frequently fail to accurately capture these diminutive, elongated features. In contrast, semantic segmentation yields fine-grained output, making it more effective for the detection of power lines under diverse backgrounds and varying illumination conditions.

4.2.1. UNet Architecture

The UNet architecture, introduced in [62] was originally developed for biomedical image segmentation; however, its structural design has demonstrated exceptional efficacy across various pixel-wise segmentation tasks. The U-Net is a convolutional neural network distinguished by its encoder-decoder configuration, which results in a symmetric “U”-shaped architecture Figure 4.1.

The encoder, also referred to as the contracting path, decreases the spatial resolution of the input image through a series of convolutional operations and max pooling layers. This process squeezes the image into a compact, feature-dense representation that encapsulates high-level semantic information. Conversely, the decoder, or expanding path, mirrors the encoder structure and is tasked with the

incremental restoration of spatial resolution through the utilization of transposed convolutions (up-convolutions).

The key aspect of the UNet architecture is use of skip connections that interlink each layer of the encoder with its respective decoder layer by concatenating their feature maps. This structural feature ensures the recovery of spatial details that may be lost during the downsampling phase, which is particularly critical for the segmentation of narrow and subtle structures such as power lines.

The U-Net architecture is particularly effective in managing small datasets, rendering it particularly well-suited for applications where annotated data is limited. This particular advantage holds significant relevance in our context, given the limited availability of annotated aerial power line datasets.

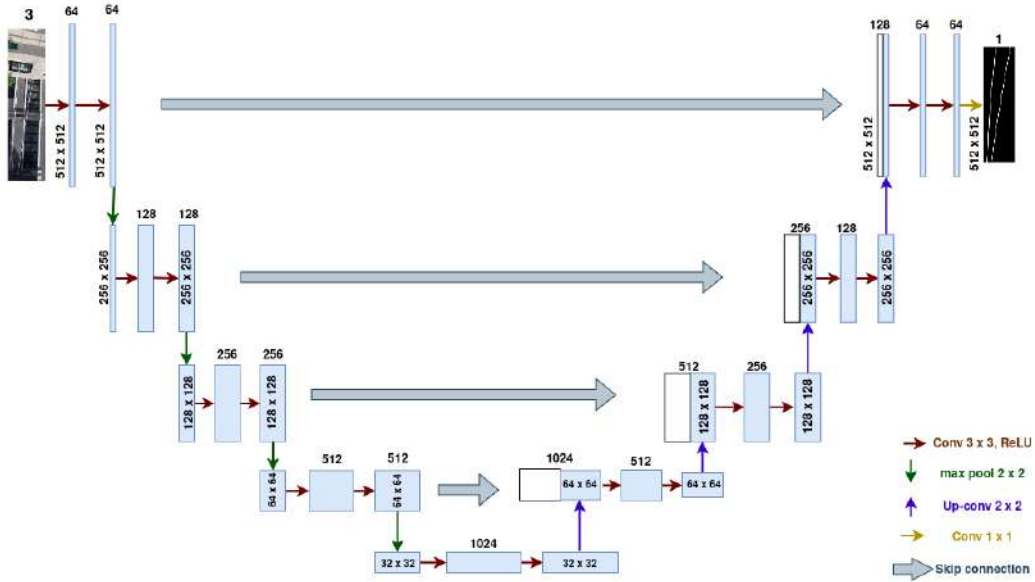


Figure 4.1: UNet Architecture

Encoder (Contracting Path)

The encoder (contracting) path is responsible for the extraction of hierarchical features from the input image, which is essential for the precise identification of the slender and elongated structure of power lines. It is comprised of multiple stages, each encompassing convolutional blocks alongside a pooling operation. The key components in each stage are:

- **Convolutional blocks:** Each encoder stage starts with two consecutive 3×3 convolutional layers, subsequently followed by a Rectified Linear Unit (ReLU) non-linear activation function. The convolutional layers serve to

extract local features while expanding the receptive field, which facilitates the network's ability to discern both fine-grained structures and contextual patterns surrounding power lines.

- **Max pooling operation:** Following each convolutional block, a 2×2 max pooling layer with a stride of 2 is employed to reduce the height and width of the feature maps by half. This downsampling operation allows the network to progressively acquire more global and abstract context, which is paramount for comprehending the potential locations of power lines in relation to the broader scene (e.g., poles, edifices, or sky regions).
- **Channel doubling and feature extraction:** With every downsampling stage, the number of feature channels is effectively doubled after each pooling step. For instance, if the initial stage outputs 64 channels, the subsequent stage would output 128, then 256, and so forth. This increase in channels mitigates the loss of spatial information by allowing the network to acquire a more extensive array of feature detectors at lower resolutions. In the context of power line segmentation, this progression enables deeper layers to capture more abstract representations such as line continuity, intersection points, or structure in cluttered urban backgrounds.

Each encoder stage, therefore, extracts features at varying scales: earlier stages concentrate on fine, localized edges (e.g., wire contours), while deeper stages encapsulate more abstract patterns and object-level context that facilitate the disambiguation of thin power lines from similar background textures. All these features will subsequently be utilized by the decoder to reconstruct the segmentation.

Bottleneck (Bridge)

The bottleneck, or bridge, serves as the connection between the encoder and decoder. It is the deepest part of the network, immediately after the final max-pooling operation, and is composed of two padded 3×3 convolutional layers accompanied by ReLU activations. In this configuration, the network processes the feature maps with the smallest spatial resolution ($1/16$ of the input) and the largest number of channels (1024).

At this stage, the network captures the most abstract, global-level information present in the image. For power line segmentation, the bottleneck empowers the model to reason regarding the overall spatial arrangement of the scene and to

learn high-level cues that differentiate wire-like structures from similar visual background textures. These deep features form a compressed yet informative representation that is subsequently employed by the decoder for the reconstruction of the segmentation mask.

Decoder (Expanding Path)

The decoder's (expansion) pathway mirrors the encoder but in reverse; it is responsible for upsampling the feature maps back to restore them to their original image resolution, consequently yielding the final segmentation output. Each stage of the decoder encompasses the following operations:

- **Transposed convolutions (up-convolutions):** Each stage within the decoder begins with a 2×2 transposed convolution that effectively doubles the spatial dimensions of the feature map. This upsampling process counteracts the spatial reduction imposed by max-pooling and progressively restores the image resolution. Furthermore, the up-convolution operation halves the number of feature channels (thereby reversing the doubling effect from the encoder), which reduces the computational load while concentrating on the enhancement of features.
- **Feature map concatenation (skip connections):** After each upsampling operation, the decoder concatenates the upsampled feature map with its corresponding feature map from the encoder (at the same resolution level). These skip connections constitute a key aspect of the U-Net architecture, facilitating the reutilization of high-resolution features from the encoder within the decoder. The reintroduction of this information via skip connections improves the decoder's capability to accurately localize thin and fine structures.
- **Convolutional blocks for feature refinement:** Following the concatenation process, the decoder employs two 3×3 convolutional layers with ReLU activations (identical to the encoder blocks) to refine the combined feature maps. This mechanism enables the network to integrate both high-level contextual information from the upsampled pathway and fine details from the encoder, resulting in precise and detailed segmentation outputs at each level of resolution. This marks the completion of one decoder stage. The procedure then repeats: the refined features undergo upsampling in the next stage, are concatenated with the encoder features, and are convolved

again.

Consequently, the decoder incrementally reconstructs the segmentation map, layer by layer, transitioning from coarse global features to fine, localized predictions, an essential aspect for accurately identifying the precise location and shape of power lines.

Skip Connections

Skip connections represent a fundamental characteristic of U-Net. They establish a connection between each encoder block and its corresponding decoder block at the same level of spatial resolution. Skip connections perform two important functions:

- **Preserving spatial resolution:** As max-pooling diminishes resolution, certain spatial details of the input may be lost during the encoding process. Skip connections facilitate the reintroduction of these details by directly transmitting the high-resolution feature maps from the encoder to the decoder. This ensures that the decoder consistently has access to both the global context derived from the bottleneck and the specific, local features from earlier layers. Consequently, this ensures that even the most diminutive, wire-like features can be accurately recovered in the final output.
- **Improving segmentation of fine structures:** Through the fusion of multi-scale information, skip connections assist the network in segmenting narrow structures that would otherwise pose challenges for resolution. For instance, in the domain of power line detection, skip connections enable the model to accurately capture the slender, elongated forms of wires by combining the high-level understanding of where lines should be with the low-level detail of edges from the encoder. Without the skip connections, the decoder's upsampled features alone may prove too coarse to delineate such fine objects with precision.

Output Layer

After the final decoder stage, the network employs a 1×1 convolution to generate the binary segmentation mask:

- **Final 1×1 convolution:** The last decoder output (comprising 64 channels) is passed through a 1×1 convolution which reduces it to a single output

channel. This layer produces per-pixel class scores, with each score indicative of the probability that the corresponding pixel is associated with a power line.

- **Binary segmentation output:** In the context of power line segmentation, the output is a single-channel probability map. These logits are converted into pixel-wise probabilities using a sigmoid activation function. Subsequently, a fixed threshold (e.g., 0.5) is applied to produce the definitive binary mask, whereby each pixel is categorized as either "power line" or "background."

Loss Function

The segmentation model is optimized utilizing a composite loss function, BCEDiceLoss, which combines Binary Cross-Entropy (BCE) and Dice loss.

1. **Binary Cross-Entropy (BCE) Loss:** BCE represents the conventional pixel-wise loss function employed for binary segmentation. It quantifies the divergence between the predicted probability distribution and the corresponding ground-truth binary mask. The BCE loss imposes a logarithmic penalty for every misclassified pixel. BCE loss is defined as:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \quad (4.1)$$

Formally, for N pixels, y_i represents the ground truth label and p_i represents the predicted probability.

2. **Dice Loss:** The Dice coefficient (Sørensen–Dice index) measures the overlap between the predicted segmentation and the actual segmentation. It is defined as:

$$\text{Dice}(P, G) = \frac{2|P \cap G|}{|P| + |G|} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}, \quad (4.2)$$

where P represent the predictions and G represents ground-truth. TP, FP, FN are true positives, false positives, and false negatives, respectively.

The Dice loss is defined as one minus the Dice coefficient (frequently incorporating a minor smoothing term, denoted as ϵ , to mitigate the risk of division by zero). Die loss is computed as:

$$\mathcal{L}_{\text{Dice}} = 1 - \text{Dice} = 1 - \frac{2 \sum_i p_i g_i + \epsilon}{\sum_i p_i + \sum_i g_i + \epsilon} \quad (4.3)$$

where $p_i, g_i \in 0, 1$ represents the predicted and true labels of pixel i .

The BCE–Dice loss integrates both components, typically formulated as a weighted summation:

$$\mathcal{L}_{\text{BCE-Dice}} = \alpha \mathcal{L}_{\text{BCE}} + (1 - \alpha) \mathcal{L}_{\text{Dice}} \quad (4.4)$$

where $\alpha \in [0, 1]$ serves as a balancing parameter, often taking a value of $\alpha = 0.5$ to ensure equal weighting. In practical implementations, the two losses are commonly aggregated through simple summation ($\alpha = 0.5$). The underlying rationale is that BCE facilitates precise probability estimations at the pixel level, whereas Dice enhances the overall spatial overlap.

4.3. Model Training

The model training procedure was executed utilizing Google Colab, employing a T4 GPU in conjunction with an Intel Core i5 CPU. The model was implemented in PyTorch, with a training pipeline tailored to ensure both convergence stability and generalization efficacy. Training was conducted with a batch size of 8 and an initial learning rate set at 0.001. The Optimization was performed utilizing the AdamW optimizer, which incorporates decoupled weight decay regularization (with a coefficient of 1×10^{-4}) to mitigate the risk of model overfitting.

To further enhance the model learning, a ReduceLROnPlateau scheduler was incorporated into the training pipeline. This scheduler monitors the validation loss and reduces the learning rate by a factor of 0.5 in instances when no improvement is detected for five consecutive epochs, with a minimum permissible learning rate of 1×10^{-6} . The number of epochs for model training was set as 70 epochs, with early stopping applied should the validation loss fail to improve over 5 successive epochs. Throughout the training, both pixel-wise accuracy and loss metrics were monitored across the training and validation datasets. The model state corresponding to the minimal validation loss was preserved and reinstated following the completion of training.

4.4. Monocular Vision-based Distance Estimation

Monocular vision refers to the utilization of a single camera for inferring spatial and depth information from sequences of images. In contrast to stereo vision systems that depend on triangulation from two viewpoints, monocular methods

depend on motion parallax, geometric constraints, and known camera intrinsics for estimating the scene structure. Within this methodology, we employ monocular vision for range estimation of power lines by analyzing their projections across sequential video frames.

A major challenge in monocular-based distance estimation lies in the lack of absolute scale. Although the direction of motion and structure can be inferred to a scale factor, the recovery of metric distances requires either scene priors, additional sensors (for example, Inertial Measurement Unit (IMU)), or known baselines. In our pipeline, we address the scale ambiguity through the integration of IMU data with monocular relative motion estimation.

4.4.1. 2D-3D Reconstruction

The fundamental concept underlying the proposed reconstruction pipeline is that a 2D line observed within a single image corresponds to a 3D plane in space when back-projected through the camera’s center of projection. By observing the same line from two different viewpoints, the associated 3D planes, each originating from a 2D observation, intersect along a unique 3D line. This intersection represents the actual power line in three-dimensional space. Exploiting this geometric relationship facilitates the reconstruction of 3D structure from monocular input, without requiring stereo vision or depth sensors.

The proposed pipeline for estimating the spatial geometry and distance of power lines from a mobile camera is comprehensively illustrated in Figure 4.2. This system leverages motion parallax induced by camera movement, in conjunction with IMU data, to derive depth information exclusively from 2D image sequences.

The pipeline begins with the extraction of binary masks using the U-Net segmentation model. These masks are then skeletonized to determine the centerlines, from which individual 2D power lines are extracted and described. Corresponding power lines are matched across successive frames utilizing spatial and geometric consistency criteria. After the matching of 2D line pairs, each line is fitted with a 2D line equation and subsequently back-projected into a 3D plane utilizing the camera’s intrinsic matrix. These planes are then transformed to express both planes within a unified reference frame, after which their intersection yields a 3D line representing the physical power line in spatial coordinates. Ultimately, the shortest distance from the camera center to the reconstructed 3D line is computed utilizing geometric distance formulas.

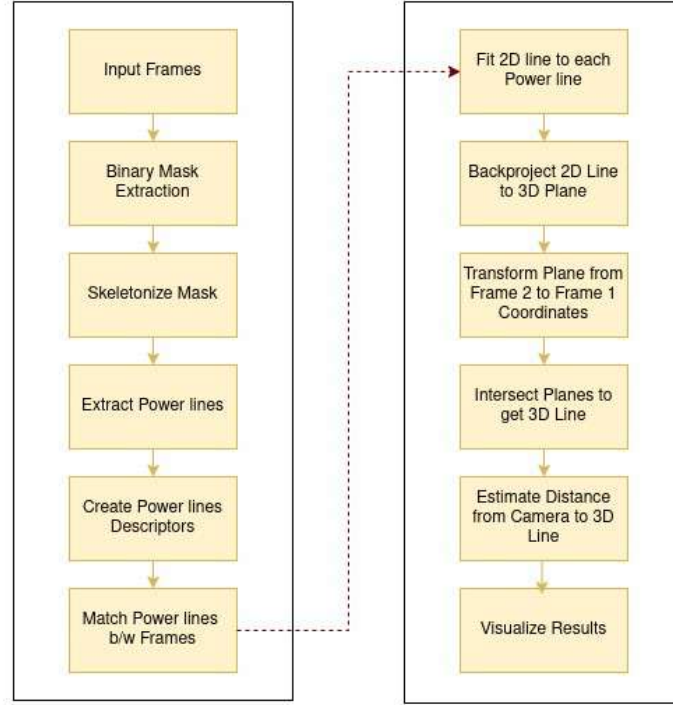


Figure 4.2: Geometric Reconstruction Pipeline

The steps of this pipeline are elaborated in the subsequent subsections.

Skeletonization

The binary mask M produced from segmentation is often characterized by substantial thickness and may encompass extensive linear regions. In order to transform each identified powerline into a representation that is confined to a single pixel in width, we implement morphological thinning, (skeletonization). This operation systematically eliminates outer pixels while maintaining the connectivity and topology of the shape, ultimately resulting in a skeleton image S . The resulting image S binary representation wherein each power line appears as a slender curve composed of pixels.

Connected Component Extraction

From the skeleton image S , we extract distinct line segments by identifying connected components. The OpenCV function `cv2.connectedComponents()` [63] is employed on S , which assigns a component ID to each pixel. For every nonzero label i , we collect the set of pixel coordinates $(x_{i,j}, y_{i,j})_{j=1}^{N_i}$ that are associated with

that component, representing one potential line segment. Components that are too small (below a pixel count threshold) may be excluded out to remove noise. The result of this step is a list of isolated line segments L_i in image coordinates, each characterized by its corresponding set of skeleton pixels.

Descriptor Construction

For each connected segment L_i , we compute a geometric descriptor to facilitate matching process between frames. The descriptor encompasses:

- **Center:** It is computed as the mean coordinate $(\bar{x}_i, \bar{y}_i)$ of the segment:

$$\bar{x}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_{i,j}, \quad \bar{y}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} y_{i,j}. \quad (4.5)$$

- **Endpoints:** Approximate endpoints $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$ of the segment, are computed as the two farthest skeleton points.
- **Orientation:** Given the line direction in radians, the angle θ_i of the segment, computed as:

$$\theta_i = \arctan 2(y_{i,2} - y_{i,1}, x_{i,2} - x_{i,1}), \quad (4.6)$$

- **Length:** The length is simply Euclidean distance between the endpoints:

$$L_i = \sqrt{(x_{i,2} - x_{i,1})^2 + (y_{i,2} - y_{i,1})^2}. \quad (4.7)$$

These descriptors summarize the position, orientation, and size of each line segment.

Powerline Matching

Given the descriptors for segments in two distinct frames (denoted frame 1 and frame 2), we match segments between frames by comparing the similarity of the descriptors. After the initial matching phase, we employ RANSAC [64] to effectively filter out spurious correspondences. This stage outputs a set of line correspondences $(L_i^{(1)}, L_j^{(2)})$ between the two frames.

Line Fitting

For each matched line segment within each frame, we fit a 2D straight line using robust regression techniques. The pixel coordinates of the segment $(x_{k,j}, y_{k,j})$ (from

frame $k = 1$ or 2) are used as data points. The `sklearn.linear_model.RANSACRegressor` [65] is utilized for model fitting:

$$y = a_k x + b_k \quad (4.8)$$

RANSAC identifies inlier points along the segment and fits a line that minimizes the the error. The resultant output comprises line parameters (a_k, b_k) corresponding to frame k . We then convert this representation into homogeneous line form,

$$\ell_k = [a_k, -1, b_k]^\top, \quad (4.9)$$

so that a point (x, y) on the line satisfies $a_k x - y + b_k = 0$. Subsequently, we normalize ℓ_k so that the normal $[a_k, -1]$ possesses a unit length. This homogeneous representation will be employed for backprojection.

Backprojection to 3D Plane

Each 2D line ℓ_k represented in image coordinates corresponds to a plane π_k in the camera-centric 3D space that encompasses the camera center as well as the 3D line. Given the camera intrinsic matrix K , the line ℓ_k in homogeneous image coordinates can be back-projected onto a plane via

$$\pi_k = K^{-T} \ell_k, \quad (4.10)$$

The intrinsic matrix K is known from camera calibration.

Pose Extraction and Relative Transformation

The absolute camera poses for the two frames are obtained from the ZED2 visual-inertial odometry (VIO) output. Each frame is associated with a quaternion q_i and translation $\mathbf{t}_i$ vector, which are subsequently converted to a rotation matrix R_i and translation vector $\mathbf{t}_i$. These represent the transformation between the camera and the world for each frame $i = 1, 2$. We then compute the relative pose from frame 1 to frame 2 as:

$$R_{\text{rel}} = R_2 R_1^\top, \quad \mathbf{t}_{\text{rel}} = \mathbf{t}_2 - R_{\text{rel}} \mathbf{t}_1. \quad (4.11)$$

This gives the rotation and translation that map coordinates between frames.

Plane Transformation

Both planes are expressed in a common coordinate frame (frame 1). The plane π_1 from frame 1 is inherently represented in the coordinates of frame 1. For the plane π_2 derived from frame 2, the relative pose is applied to transform it into frame 1. If $\pi_2 = [\mathbf{n}_2^\top, d_2]^\top$ represents the plane in frame 2, then its parameters in frame 1 are given by:

$$\mathbf{n}_2^{(1)} = R_{\text{rel}}^\top \mathbf{n}_2, \quad d_2^{(1)} = d_2 + \mathbf{n}_2^\top \mathbf{t}_{\text{rel}}. \quad (4.12)$$

Here, $\mathbf{n}_2^{(1)}$ is the rotated normal vector, and the offset is adjusted by the translation component along the normal. Consequently, we obtain two plane equations expressed in the coordinates of frame 1:

$$\pi_1 : \mathbf{n}_1^\top X + d_1 = 0, \quad \pi_2^{(1)} : \left(\mathbf{n}_2^{(1)} \right)^\top X + d_2^{(1)} = 0 \quad (4.13)$$

3D Line Intersection

The intersection of the two planes π_1 and $\pi_2^{(1)}$ forms a 3D line. The direction vector of this line is determined by the cross product of the plane normals:

$$\mathbf{v} = \mathbf{n}_1 \times \mathbf{n}_2^{(1)}. \quad (4.14)$$

To find a point $\mathbf{P}$ on this line, we solve a system of linear equations formed by combining the two plane equations with an additional constraint that selects a unique point closest to the origin.

$$A = \begin{bmatrix} \mathbf{n}_1^\top \\ \left(\mathbf{n}_2^{(1)} \right)^\top \\ \mathbf{v}^\top \end{bmatrix}, \quad \mathbf{b} = - \begin{bmatrix} d_1 \\ d_2^{(1)} \\ 0 \end{bmatrix} \quad (4.15)$$

Then, solving $A\mathbf{P} = \mathbf{b}$ for finding one particular point $\mathbf{P}$ satisfying both planes. The reconstructed 3D power line is parametrically represented as $\mathbf{X}(t) = \mathbf{P} + t \cdot \mathbf{v}$ in the frame-1 coordinate system.

Distance Computation

To compute the distance between the camera and the reconstructed 3D line (power line), the perpendicular distance from the camera center to the line is calculated. Given a point $\mathbf{P}$ on the line and direction vector $\mathbf{v}$, the distance is expressed as:

$$d = \frac{\|(\mathbf{P} - \mathbf{O}) \times \mathbf{v}\|}{\|\mathbf{v}\|} \quad (4.16)$$

where $\mathbf{O} = (0,0,0)^\top$ denotes the camera center. given $\mathbf{O}$ is the origin, this expression simplifies to $\|\mathbf{P} \times \mathbf{v}\|/\|\mathbf{v}\|$. This distance d represents the shortest distance from the UAV to the power line, measured perpendicular to the line's direction.

4.5. Integrated Framework for Power Line Distance Estimation

To estimate the 3D geometry of power lines from monocular image sequences, the segmentation and 3D reconstruction modules were integrated into a unified framework 2. The input to the pipeline is a sequence of images, the associated camera intrinsics, and pose estimates for each frame, subsequently yielding three-dimensional reconstructions of power lines alongside their respective distances from the camera.

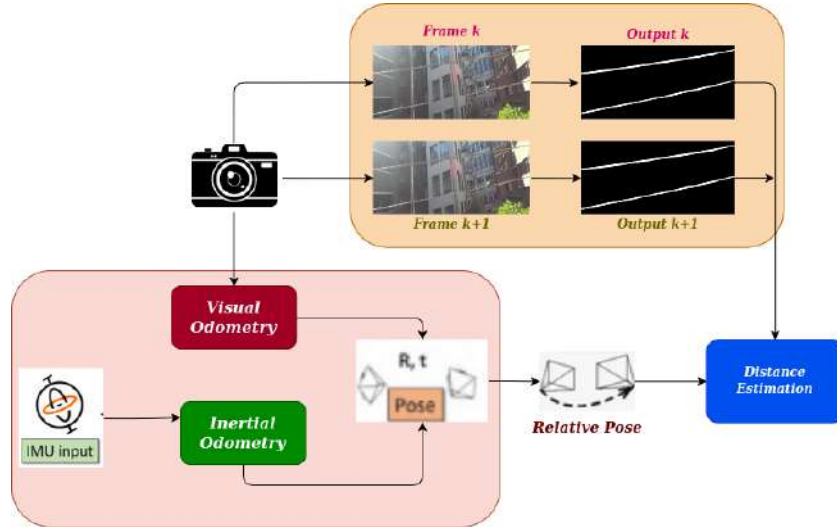


Figure 4.3: Power Line Distance Estimation Pipeline

The pipeline Figure 4.3 begins with the initialization of the pre-trained UNet model for the semantic segmentation of power lines. For each pair of frames within the image sequence, the UNet model generates binary segmentation masks that accurately delineate the power lines. These segmentation masks are post-processed to extract skeletonized representations. The skeletonized representations of the power lines are subsequently matched utilizing a robust matching algorithm. After

correspondences are established, 2D line equations are fitted to the matched line segments in both frames. These 2D lines are then backprojected into 3D space as planes by employing the known camera intrinsic matrix. Leveraging the relative pose (comprising rotation matrix and translation vector) between the two frames, the plane derived from the second frame is transformed into the coordinate system of the first frame. The 3D power line is then reconstructed by computing the intersection of the two planes. This results in a 3D line that represents the actual spatial trajectory of the observed power line segment. The perpendicular distance from the camera to the reconstructed 3D line is computed to estimate power line proximity.

Algorithm 2 Power Line 3D Estimation Pipeline

```

1: Input: Image sequence, camera intrinsics  $K$ , poses
2: Output: 3D power line reconstructions with distances
3: procedure MAINPIPELINE
4:   Load UNet segmentation model
5:   Load image sequence and corresponding poses
6:   for each consecutive image pair  $(I_t, I_{t+1})$  do
7:     Segment power lines using UNet:  $S_t, S_{t+1} \leftarrow \text{UNet}(I_t), \text{UNet}(I_{t+1})$ 
8:     Extract power line skeletons:  $\mathcal{P}_t, \mathcal{P}_{t+1} \leftarrow \text{ProcessPowerLines}(S_t, S_{t+1})$ 
9:     Match power lines between frames:  $\mathcal{M} \leftarrow \text{MatchPowerLines}(\mathcal{P}_t, \mathcal{P}_{t+1})$ 
10:    Compute relative pose  $(R, t)$  between frames
11:    for each matched pair  $(p_i, p_j) \in \mathcal{M}$  do
12:      Fit 2D line equations:  $\ell_i, \ell_j \leftarrow \text{FitLine}(p_i), \text{FitLine}(p_j)$ 
13:      Backproject to 3D planes:  $\pi_i \leftarrow K^{-T} \ell_i, \pi_j \leftarrow K^{-T} \ell_j$ 
14:      Transform  $\pi_j$  to frame  $t$ 's coordinate system
15:      Compute 3D line as intersection:  $L \leftarrow \pi_i \cap \pi_j$ 
16:      Estimate distance from camera to 3D line
17:      Store 3D line parameters and distance
18:    end for
19:  end for
20:  Output all 3D power line reconstructions and distances.
21: end procedure

```

4.6. Evaluation Metrics

For evaluating the performance of the proposed methodology for detecting the power lines and estimating their distances, we utilized a set of standard evaluation metrics. The performance of the UNet model is evaluated using standard classification-based metrics: Accuracy, Precision, Recall, and F₁ Score, previously defined in (Table 3.2). Furthermore, the overlap-based metric is also introduced:

- **Intersection over union:** IOU measures the overlap between predicted and ground truth foreground regions. It serves as a more stringent overlap metric that imposes penalties for both false positives and false negatives.

$$\frac{TP}{TP + FP + FN} \quad (4.17)$$

Additionally, Table 4.1 presents the metrics used for assessing the accuracy and consistency of distance estimations.

Table 4.1: Evaluation metrics used for assessing distance estimation performance.

Metric	Description	Formula
MAE	Average magnitude of errors, ignoring direction. Expressed in the same units as the data.	$MAE = \frac{1}{N} \sum_{i=1}^N d_i - \hat{d}_i $
RMSE	Root mean of squared errors; penalizes larger errors more heavily. Common in 3D reconstruction.	$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (d_i - \hat{d}_i)^2}$
Std Dev	Measures error spread ($e_i = d_i - \hat{d}_i$). Low values indicate consistent predictions.	$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (e_i - \bar{e})^2}$

Where: d_i = ground truth, $\hat{d}_i$ = prediction, $e_i = d_i - \hat{d}_i$, $\bar{e}$ = mean error, N = number of samples.

4.7. Results

This section presents the qualitative and quantitative results obtained from the proposed methodology for power line segmentation and distance estimation.

The U-Net based segmentation model was evaluated utilizing a test dataset comprising a variety of outdoor scenes that prominently feature power lines against heterogeneous backgrounds. Table 4.2 encapsulates the quantitative performance metrics. The model demonstrated a high accuracy in the power-line segmentation task, attaining an overall pixel accuracy of 0.9951. However, as is widely acknowledged, accuracy can be a deceptive metric when the power line constitutes a very small fraction of the image. In such instances, overlap-based metrics, such as the F_1 score and the Intersection-over-Union (IoU), provide a more nuanced understanding. The F_1 score of our model is 0.8976, and its IoU is 0.8142. These values suggest that the predicted line mask exhibits a substantial alignment with the ground truth: approximately 81.4% of the union between predicted and actual line pixels coincides, and the harmonic mean of precision and recall approaches 0.90.

In this context, the precision of 0.9185 indicates that of all pixels classified as “line,” roughly 91.85% were indeed true positives, whereas the recall of 0.8776 signifies that approximately 87.76% of actual line pixels were accurately identified. In other terms, the model tends to be slightly conservative: it occasionally misses thin line segments (false negatives) more often than it fabricates spurious lines (false positives). The relatively minor discrepancy (precision > recall) implies that the majority of errors are minor omissions rather than significant misrepresentations.

Table 4.2: UNet Segmentation Model Performance Metrics for Power Lines

Metric	Accuracy	Precision	Recall	F1 Score	IoU
Value	0.9951	0.9185	0.8776	0.8976	0.8142

Qualitative results are shown in Figure 4.4, which includes the original input images, their corresponding ground truth masks, and the predicted masks from the model. The predicted mask (right panel) demonstrates a high degree of fidelity to the ground truth (center panel), covering nearly all continuous wire segments. A few discrepancies appear as very slight thickening of the line or minor gaps in less pronounced sections. These visual distinctions align precisely with the obtained precision and recall metrics: a few true line pixels are omitted (thereby reducing recall), and a few extraneous pixels are incorporated around the line (thereby diminishing precision). Collectively, these results affirm the model’s proficiency in effectively segmenting power lines.

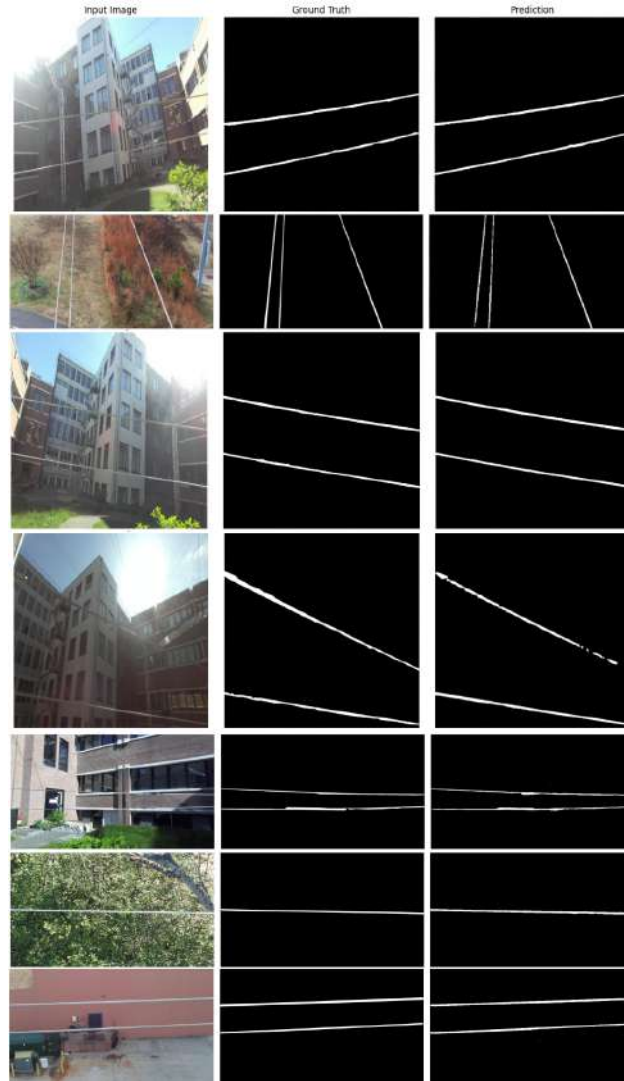


Figure 4.4: U-Net model visualization for Power line segmentation: columns display from left to right: the original image, ground truth masks, and model predictions

In order to evaluate the accuracy of the distance estimation pipeline, data was captured by marking actual ground truth distances of 2, 3.5, and 7 meters from a power line utilizing a measuring scale. An individual operating the ZED2 stereo camera (Only the left frames were taken to replicate the monocular camera) followed a trajectory parallel to the power line at these designated distances while acquiring image sequences. This setup provided reliable ground truth values for quantitative evaluation of the estimated distances.

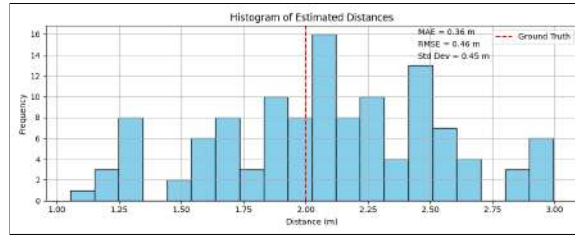
The distance estimation results obtained from test sequences are shown in 4.5, 4.6, 4.7. The histograms of the estimated distances indicate that the system consistently produces values that are close to the corresponding ground truth mea-

surements, exhibiting a moderate dispersion that reflects the inherent difficulties associated with monocular reconstruction. The associated error histogram exhibits a slight positive skew, reflecting occasional minor overestimations; however, the majority of estimations remain within acceptable limits for practical applications.

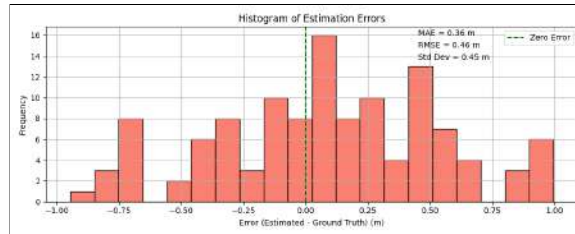
Table 4.3: Distance estimation performance at different ground truth distances.

Ground Truth (m)	MED (m)	MAE (m)	RMSE (m)	Std Dev (m)
2.0	2.0847	0.3618	0.4567	0.4487
3.5	3.4508	0.2609	0.3083	0.3044
7.0	6.9862	0.2177	0.2625	0.2622

The error metrics Table 4.3 across all scenarios demonstrate the system’s reliability. The Mean Absolute Error (MAE) ranges from 0.22 m to 0.36 m, whereas the Root Mean Squared Error (RMSE) spans from 0.27 m to 0.46 m. Moreover, the standard deviation of errors fluctuates between 0.26,m and 0.45 m, indicating a moderate level of consistency in distance estimations across diverse testing conditions. These results are particularly promising, as they validate the feasibility of employing the proposed monocular reconstruction methodology for power line distance estimation in real-world outdoor settings, even in the presence of inherent challenges posed by minor IMU drift, segmentation discrepancies, and fluctuating scene conditions.



(a) Histogram of Estimated Distances



(b) Histogram of Estimated Errors

Figure 4.5: Results (Ground Truth 2m)

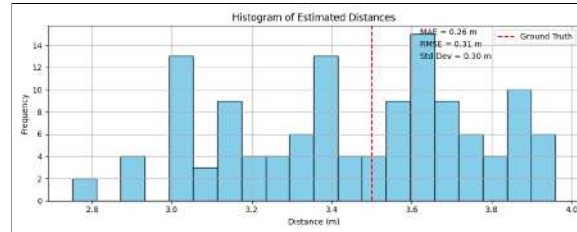


(a) Error Metrics

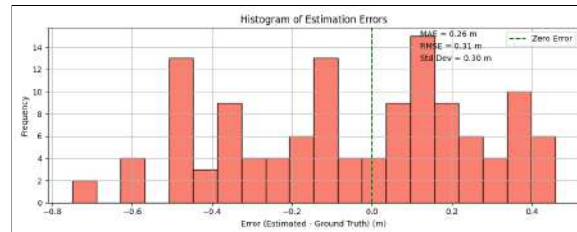
Figure 4.5: Results (Ground Truth 2m) (continued)

It is important to mention that a significant factor that influences the precision of distance estimation is the relative motion between frames. Specifically, it is crucial to achieve adequate translation and rotation (pose change) between frames to ensure that the back-projected planes from the 2D lines properly intersect. In instances where consecutive frames demonstrate minimal relative motion, the reconstructed planes tend to be nearly parallel, resulting in suboptimal intersection geometry and unreliable distance estimations. This was evident during test sequences, wherein constrained camera motion led to unstable distance estimates.

In summary, the present results indicate a reliable performance for UAV-based power line distance estimation utilizing monocular vision. Although the system

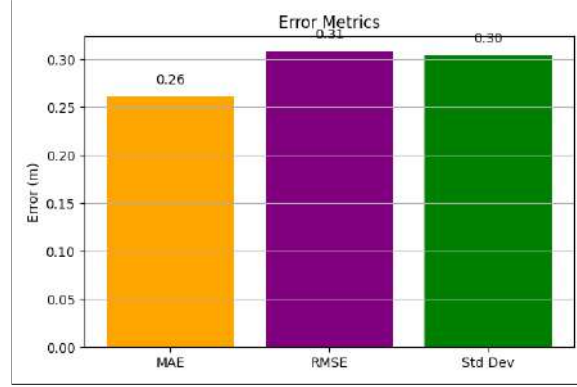


(a) Histogram of Estimated Distances



(b) Histogram of Estimated Errors

Figure 4.6: Results (Ground Truth 3.5m)

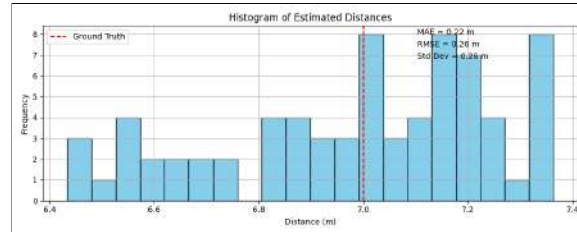


(a) Error Metrics

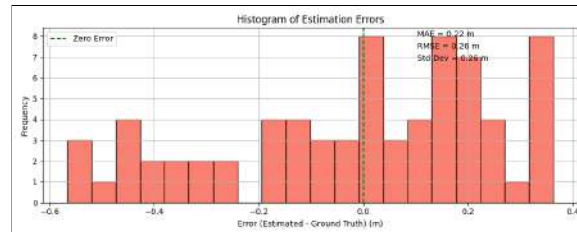
Figure 4.6: Results (Ground Truth 3.5m) (continued)

exhibits promising accuracy, further enhancements could be attained by integrating additional frame pairs for multi-view intersection consistency and implementing temporal smoothing techniques to mitigate the effects of pose estimation noise. Such improvements would likely improve both stability and precision, particularly over long distances and in challenging environments.

The developed system was deployed on NVIDIA Jetson AGX Xavier to evaluate its processing capabilities. The Jetson provides an AI performance of up to 32 TOPS, featuring an 8-core NVIDIA Armv8.2 64-bit CPU alongside a Volta GPU equipped with 512 CUDA cores and 64 Tensor cores. As shown in Figure 4.8, the system currently attains a processing time per frame of around 0.6 to 0.7 seconds.

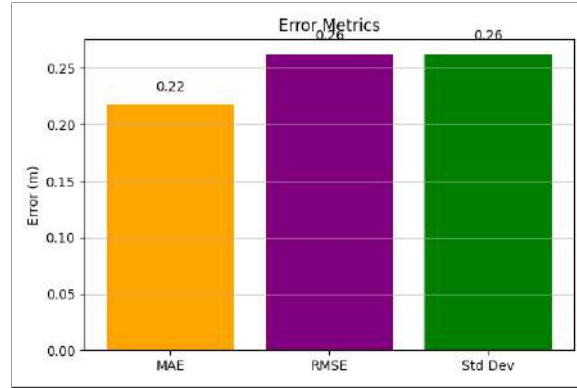


(a) Histogram of Estimated Distances



(b) Histogram of Estimated Errors

Figure 4.7: Results (Ground Truth 7m)



(a) Error Metrics

Figure 4.7: Results (Ground Truth 7m) (continued)

The operations that demand the greatest computational resources include image pre-processing and segmentation, which substantially affect the overall latency. These results suggest that although the system is capable of near real-time performance, additional optimization is necessary for deployment in high-speed drone navigation scenarios. Performance can be improved by applying TensorRT-based acceleration for model inference, quantizing the segmentation model and parallelizing CPU-bound operations. Additionally, exploring model pruning could yield further improvements without compromising detection accuracy.

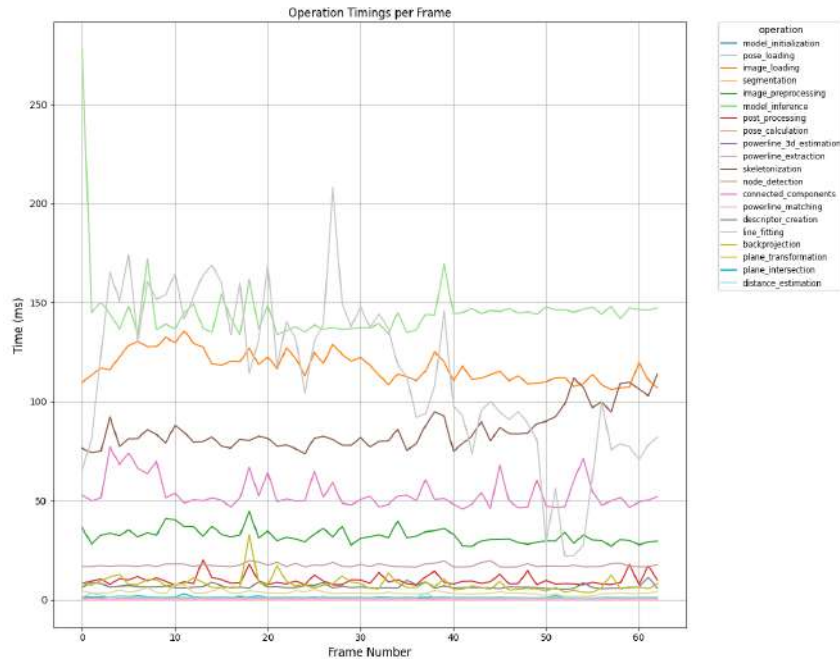


Figure 4.8: Execution time per frame for each operation in the pipeline

5. Conclusion

This thesis focused on the detection and ranging of power lines. The goal was to develop and evaluate a robust perception pipeline capable of detecting power lines and estimating their metric distance to facilitate drone navigation and obstacle avoidance. This concluding chapter encapsulates the principal findings by directly addressing the research questions and provides recommendations for future research.

5.1. Addressing the Research Questions

1. **What state-of-the-art deep learning architectures are most effective for the detection of thin objects in aerial imagery?**

Various deep learning architectures were explored, including object detection models (YOLOv5, YOLOv8, and YOLOv11) and the UNet segmentation model. The experimental results revealed that YOLOv8 surpassed both YOLOv5 and YOLOv11 in terms of precision and recall when detecting power lines. Moreover, the U-Net model, when trained with appropriate augmentations and a composite loss function that integrates Binary Cross-Entropy and Dice Loss, accomplished high-fidelity pixel-level segmentation. This allowed for a more accurate representation of slender structures.

2. **What are effective approaches for reliably estimating the distance of detected objects**

The distance estimation in this study compared both stereo and monocular vision techniques. The stereo vision relied on disparity calculations from point features extracted from rectified left and right image pairs. Detected bounding boxes from the YOLOv8 model served as regions of interest (ROIs) to limit the disparity search. Nonetheless, the precision of depth estimates was found to be acutely sensitive to the dimensions and accuracy of these bounding boxes. Specifically, oversized or inadequately fitted bounding

boxes frequently included background pixels, resulting in erroneous disparity values and unreliable distance estimates.

In the monocular vision method, power lines detected across a sequence of frames were modeled as 2D line segments. Subsequently, the 2D-to-3D reconstruction techniques were applied for the computation of distance. However, as with all monocular methods, this reconstruction solely provided relative depth information, lacking a true metric scale. To rectify this scale ambiguity, inertial measurement unit (IMU) data from the onboard sensor suite was integrated. This incorporation enabled the recovery of metric distances from monocular image sequences, thus enabling reliable distance estimation using a single moving camera.

3. **What method is the most effective for detecting and estimating the metric distance of the power lines in aerial images captured by UAVs?**

The methodology combining segmentation-based detection with 2D-to-3D reconstruction provided the most reliable results. In contrast to bounding box-based methodologies, segmentation enabled a more refined localization of the object's true structure, thus allowing for more accurate depth estimations. By employing the 2D line equations derived from segmentation masks, followed by 3D reconstruction, distance estimates were produced that closely aligned with ground truth values. Consequently, for power lines, segmentation-driven 3D reconstruction demonstrates a promising and more precise methodology for metric distance estimation in aerial imagery captured by unmanned aerial vehicles (UAVs).

5.2. Recommendations for further research

While this dissertation presents a promising methodology for the detection and localization of slender obstacles, such as power lines, several directions remain open for further exploration and enhancement:

- **Data Collection in Diverse Environments:** Future research should include collecting and curating custom datasets across varied environments to capture a broader distribution of power line appearances, lighting conditions, and background complexity. These datasets would enhance generalization and enable model performance benchmarking in underrepresented or challenging scenarios.

- **Multi-Modal Sensor Fusion for Robust Detection:** Multi-modal sensor fusion including the use of thermal data alongside the RGB data may be investigated in future studies to enhance the detection robustness under varying environmental and lighting conditions. Thermal cameras provide visibility in low-light or high-contrast scenes, in which the performance of RGB modalities may diminish. This fusion can improve the model's ability to generalize better to challenging real-world scenarios, such as fog, dusk, or highly reflective backgrounds.
- **Robust Visual-inertial Odometry (VIO):** The efficacy of monocular 3D reconstruction heavily depends on the precise estimation of the camera pose (rotation and translation). Future work should consider developing or integrating robust pose estimation techniques. Moreover, a closer integration with IMU data, in conjunction with the implementation of optimization techniques such as bundle adjustment or factor graph optimization could enhance the stability and metric consistency of the reconstruction pipeline.
- **Real-World Deployment and Validation:** A key next step is the deployment of the developed pipeline on embedded UAV platforms for field testing. Real-world testing under various conditions would provide valuable information on the practical performance and robustness of the system.

BIBLIOGRAPHY

- [1] R. Khanam and M. Hussain, “What is yolov5: A deep look into the internal features of the popular object detector,” *arXiv preprint arXiv:2407.20892*, 2024.
- [2] T. Cabreira, L. Brisolara, and P. F. Jr., “Survey on coverage path planning with unmanned aerial vehicles,” *Drones*, vol. 3, no. 1, p. 4, 2019.
- [3] A. Merei, H. Mcheick, and A. Ghaddar, “Survey on path planning for uavs in healthcare missions,” *Journal of Medical Systems*, vol. 47, no. 1, p. 79, 2023.
- [4] A. Ghaddar, A. Merei, and E. Natalizio, “Pps: Energy-aware grid-based coverage path planning for uavs using area partitioning in the presence of nfzs,” *Sensors*, vol. 20, no. 13, 2020.
- [5] L. Collins, P. Ghassemi, E. T. Esfahani, D. Doermann, K. Dantu, and S. Chowdhury, “Scalable coverage path planning of multi-robot teams for monitoring non-convex areas,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9777–9783, IEEE, 2021.
- [6] H. Wang, S. Zhang, X. Zhang, X. Zhang, and J. Liu, “Near-optimal 3-d visual coverage for quadrotor unmanned aerial vehicles under photogrammetric constraints,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 2, pp. 1694–1704, 2022.
- [7] C. B. Kamatchi and A. Muthukumaravel, “Enhancing maize cultivation with intelligent decision support: Iot, deep learning, and blockchain integration for land data analysis,” in *2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT)*, pp. 1–5, IEEE, 2024.
- [8] P. Shobeiry, “Uav path planning for wildfire tracking using partially observable markov decision process,” in *Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 123–130, IEEE, 2021.

- [9] A. Merei, H. Mcheick, A. Ghaddar, and D. Rebaine, “A survey on obstacle detection and avoidance methods for uavs,” *Drones*, vol. 9, no. 3, 2025.
- [10] B. Li, Z. Fei, and Y. Zhang, “Uav communications for 5g and beyond: Recent advances and future trends,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2241–2263, 2019.
- [11] C. Zhang, X. Weng, Y. Cao, and M. Ding, “Monocular absolute depth estimation from motion for small unmanned aerial vehicles by geometry-based scale recovery,” *Sensors*, vol. 24, no. 14, 2024.
- [12] H. Yu, F. Zhang, P. Huang, C. Wang, and L. Yuanhao, “Autonomous obstacle avoidance for uav based on fusion of radar and monocular camera,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5954–5961, 2020.
- [13] Z. Peng, J. Wu, Y. Huang, Z. Liang, T. Sun, and Z. Li, “Deep learning-based power line corridor intrusion detection,” in *2024 6th International Conference on Robotics and Computer Vision (ICRCV)*, pp. 115–120, 2024.
- [14] R. Abdelfattah, X. Wang, and S. Wang, “Ttpla: An aerial-image dataset for detection and segmentation of transmission towers and power lines,” in *Proceedings of the Asian Conference on Computer Vision*, 2020.
- [15] B. Mendu and N. Mbuli, “State-of-the-art review on the application of unmanned aerial vehicles (uavs) in power line inspections: Current innovations, trends, and future prospects,” *Drones*, vol. 9, no. 4, 2025.
- [16] J. Zhao, Q. Zhao, C. Wu, Z. Li, and F. Shuang, “A scale-invariant looming detector for uav return missions in power line scenarios,” *Biomimetics*, vol. 10, no. 2, 2025.
- [17] M. Lehtomäki, A. Kukko, L. Matikainen, J. Hyypä, H. Kaartinen, and A. Jaakkola, “Power line mapping technique using all-terrain mobile laser scanning,” *Automation in Construction*, vol. 105, p. 102802, 2019.
- [18] H. B. Kim and G. Sohn, “Random forests based multiple classifier system for power-line scene classification,” in *ISPRS Calgary 2011 Workshop*, vol. XXXVIII-5/W12, pp. 253–258, 2011.

- [19] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang, "Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform," *Machine Vision and Applications*, vol. 21, no. 5, pp. 677–686, 2010.
- [20] H. Sharma, R. Bhujade, A. V., and B. P., "Vision-based detection of power distribution lines in complex remote surroundings," in *2014 IEEE International Conference on Image Processing (ICIP)*, 2014.
- [21] J. Song, J. Qian, Y. Li, Z. Liu, Y. Chen, and J. Chen, "Automatic extraction of power lines from aerial images of unmanned aerial vehicles," *Sensors*, vol. 22, no. 17, p. 6431, 2022.
- [22] T. Santos, T. Cunha, A. Dias, A. P. Moreira, and J. Almeida, "Uav visual and thermographic power line detection using deep learning," *Sensors*, vol. 24, no. 17, p. 5678, 2024.
- [23] J. Gwizdała, D. Öner, S. K. Roy, M. A. Shah, A. Eberhard, I. Egorov, P. Krüsi, G. Yakushev, and P. Fua, "Vision-based power line cables and pylons detection for low flying aircraft," *arXiv preprint arXiv:2407.14352*, 2024.
- [24] V. N. Nguyen, R. Jenssen, and D. Roverso, "LS-Net: Fast single-shot line-segment detector," *arXiv preprint arXiv:1912.09532*, 2019.
- [25] A. Varghese, J. Gubbi, H. Sharma, and P. Balamuralidhar, "Power infrastructure monitoring and damage detection using drone captured images," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, pp. 207–210, 2017.
- [26] J. Choi and S. J. Lee, "Weakly supervised learning for transmission line detection using unpaired image-to-image translation," *Remote Sensing*, vol. 14, no. 14, pp. 3421–3421, 2022.
- [27] A. Stambler, G. Sherwin, and P. Rowe, "Detection and reconstruction of wires using cameras for aircraft safety systems," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 697–704, 2019.
- [28] R. Madaan, D. Maturana, and S. Scherer, "Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles," technical report, Robotics Institute, Carnegie Mellon University, 2017.

- [29] H. Aboalia, S. Hussein, and A. Mahmoud, “Enhancing power lines detection using deep learning and feature-level fusion of infrared and visible light images,” *Arabian Journal for Science and Engineering*, 2024.
- [30] E. Bellou, I. Pisica, and K. Banitsas, “Aerial inspection of high-voltage power lines using yolov8 real-time object detector,” *Energies*, vol. 17, p. 2535, May 2024.
- [31] C. Wang, Y. Chen, Z. Wu, B. Liu, H. Tian, D. Jiang, and X. Sun, “Line-yolo: An efficient detection algorithm for power line angle,” *Sensors*, vol. 25, p. 876, Jan 2025.
- [32] S. Rong, L. He, S. F. Atici, and A. E. Cetin, “Advanced yolo-based real-time power line detection for vegetation management,” *IEEE Transactions on Power Systems*, 2025. Early Access.
- [33] M. Cano-Solis, J. Ballesteros, and G. Sanchez-Torres, “Vepl-net: a deep learning ensemble for automatic segmentation of vegetation encroachment in power line corridors using uav imagery,” *ISPRS International Journal of Geo-Information*, vol. 12, no. 11, p. 454, 2023.
- [34] M. Mao, Y. Chen, W. Chen, W. Du, M. Zhang, and T. Mao, “Power transmission line image segmentation method based on binocular vision and feature pyramid network,” in *2021 International Conference on Sensing, Measurement and Data Analytics in the Era of Artificial Intelligence (ICSMD)*, (Nanjing, China), pp. 1–4, IEEE, 2021.
- [35] G. A. N. Pussente, I. Z. Biundini, A. L. M. Marcato, and E. P. de Aguiar, “Advanced drone-based powerline inspection using image segmentation and adaptive visual control,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 47, p. 112, Mar 2025.
- [36] A. Tsellou, G. Livanos, D. Ramnalis, V. Polychronos, G. Plokamakis, M. Zervakis, and K. Moirogiorgou, “A uav intelligent system for greek power lines monitoring,” *Sensors*, vol. 23, no. 20, p. 8441, 2023.
- [37] S. Li, Y. Zhang, Y. Zhang, and Y. Zhang, “Real-time power line detection network using visible light and infrared images,” in *2019 IEEE International Conference on Image Processing (ICIP)*, (Taipei, Taiwan), pp. 3886–3890, IEEE, 2019.

- [38] W. Zhao, Q. Dong, and Z. Zuo, “A method combining line detection and semantic segmentation for power line extraction from unmanned aerial vehicle images,” *Remote Sensing*, vol. 14, p. 1367, Mar 2022.
- [39] H. Zhang, W. Yang, H. Yu, H. Zhang, and G.-S. Xia, “Detecting power lines in uav images with convolutional features and structured constraints,” *Remote Sensing*, vol. 11, no. 11, p. 1342, 2019.
- [40] D. An, Q. Zhang, J. Chao, T. Li, F. Qiao, Y. Deng, and Z. Bian, “Duf-former: Solving power line detection task in aerial images using semantic segmentation,” *arXiv preprint arXiv:2304.05821*, 2023.
- [41] R. Abdelfattah, X. Wang, and S. Wang, “Plgan: Generative adversarial networks for power-line segmentation in aerial images,” *arXiv preprint arXiv:2204.07243*, 2022.
- [42] Y. Cheng, Z. Chen, and D. Liu, “PL-UNeXt: Per-stage edge detail and line feature guided segmentation for power line detection,” *arXiv preprint arXiv:2303.04413*, 2023.
- [43] G. Dubey, R. Madaan, and S. Scherer, “Droan – disparity-space representation for obstacle avoidance: Enabling wire mapping & avoidance,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6311–6318, IEEE, October 2018.
- [44] Y. Ma, Z. Yu, Y. Zhou, Q. Li, and Y. Wu, “Uav system with trinocular vision for external obstacle detection of transmission lines,” *Applied Optics*, vol. 61, no. 12, pp. 3297–3311, 2022.
- [45] X. Zhou, X. Zheng, and K. Ou, “Power line detect system based on stereo vision and fpga,” in *2017 2nd International Conference on Image, Vision and Computing*, pp. 715–719, IEEE, 2017.
- [46] Y. Zhang, X. Yuan, W. Li, and S. Chen, “Automatic power line inspection using uav images,” *Remote Sensing*, vol. 9, no. 8, p. 824, 2017.
- [47] R. Madaan, M. Kaess, and S. Scherer, “Multi-view reconstruction of wires using a catenary model,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (Montreal, Canada), pp. 5657–5664, May 2019.

- [48] C. Zhou, J. Yang, C. Zhao, and G. Hua, “Fast, accurate thin-structure obstacle detection for autonomous mobile robots,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 318–327, IEEE, July 2017.
- [49] E. Pastucha, E. Puniach, A. Ścisłowicz, P. Ćwiakała, W. Niewiem, and P. Wiącek, “3d reconstruction of power lines using uav images to monitor corridor clearance,” *Remote Sensing*, vol. 12, no. 22, p. 3698, 2020.
- [50] F. Ganovelli, L. Malomo, and R. Scopigno, “Reconstructing power lines from images,” in *Proceedings of the 2018 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, IEEE, 2018.
- [51] P. Wang, L. Liu, N. Chen, H.-K. Chu, C. Theobalt, and W. Wang, “Vid2curve: Simultaneous camera motion estimation and thin structure reconstruction from an rgb video,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 96:1–96:14, 2020.
- [52] B. Přibyl, P. Zemčík, and M. Čadík, “Absolute pose estimation from line correspondences using direct linear transformation,” *Computer Vision and Image Understanding*, vol. 161, no. 1, pp. 130–144, 2017.
- [53] S. Jiang and W. Jiang, “Uav-based oblique photogrammetry for 3d reconstruction of transmission line: Practices and applications,” *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W13, pp. 401–406, 2019.
- [54] G. Schindler, P. Krishnamurthy, and F. Dellaert, “Line-based structure from motion for urban environments,” in *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT’06)*, pp. 846–853, 2006.
- [55] H. Zhang, W. Yang, H. Yu, H. Zhang, and G.-S. Xia, “Pld-uav: Power line detection in uav,” 2019. Accessed: 2025-05-17.
- [56] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” in *Advances in Neural Information Processing Systems*, vol. 27, pp. 3320–3328, 2014.

- [57] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas,” *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.
- [58] Ultralytics, “Yolov5 architecture description.” https://docs.ultralytics.com/yolov5/tutorials/architecture_description/, 2023. Accessed: 2025-05-21.
- [59] S.-H. Tsang, “Brief review: Yolov5 for object detection.” <https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a>, 2023. Accessed: 2025-05-21.
- [60] Stereolabs, “ZED 2i Stereo Camera.” <https://www.stereolabs.com/en-nl/store/products/zed-2i>. Accessed: 2025-05-23.
- [61] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.
- [62] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, Springer, 2015.
- [63] G. Bradski, “The opencv library,” *Dr. Dobb’s Journal of Software Tools*, vol. 25, no. 11, pp. 120–125, 2000.
- [64] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” in *Communications of the ACM*, vol. 24, pp. 381–395, 1981.
- [65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-learn: Machine learning in python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

12

Urban Object Detection Using Sensor Fusion for Autonomous
Navigation
Selin Yavuz



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTER IN
INTELLIGENT FIELD ROBOTIC SYSTEMS

Urban Object Detection Using Sensor Fusion for Autonomous Navigation

Author:

Selin Yavuz

MSc Intelligent Field Robotic Systems

Internal supervisor:

Dr. Dániel Horváth

Research Fellow at ELTE

External supervisor:

Assoc. Prof. Fernando Garcia

Head of AMPL

Madrid, May 29, 2025

Abstract

Urban object detection remains an important challenge for autonomous navigation, particularly in accurately identifying and locating thin and small structures such as poles. These structures are often difficult to detect due to their size, shape, and integration into complex urban scenes.

In response to these challenges, this thesis proposes a dual-stage approach to urban object detection by integrating 2D deep learning-based object detection into 3D spatial representation. The methodology consists of two main stages: (1) 2D object detection, which involves fine-tuning a pretrained model to identify pole-like objects in urban environments, and (2) 3D object detection, where the 2D detections are projected into 3D space and further processed through point cloud clustering and outlier rejection to generate accurate 3D bounding boxes.

The approach shows how combining segmentation-derived labels with geometric modeling can serve as an alternative to fully supervised 3D object inference. The model demonstrates promising performance on a custom dataset, benefiting from the SAHI framework, which enhances the detection of smaller or more challenging-to-see objects. These results indicate that integrating 2D and 3D detection methods can provide valuable spatial information, even when detailed 3D annotations are limited or unavailable, supporting practical urban navigation applications.

This thesis presents a foundational pipeline for pole detection in urban navigation, laying the groundwork for more adaptable and robust autonomous systems in complex real-world environments. Further development, including dataset growth and model refinement, can enhance detection accuracy and robustness, thereby advancing the effectiveness of autonomous systems in urban environments.

Keywords: Urban object detection, autonomous navigation, 2D-3D projection, deep learning, sensor fusion, SAHI, pole detection

Acknowledgements

I would like to thank my supervisors sincerely, Dr. Dániel Horváth from Eötvös Loránd University (ELTE) and Associate Professor Fernando García Fernández, head of the AMPL laboratory at Universidad Carlos III de Madrid (UC3M), for their invaluable guidance and support throughout my thesis. Dr. Horváth's feedback and continuous encouragement significantly shaped the direction and quality of my research. At the same time, Associate Professor Fernando García Fernández's insightful advice and the enriching environment of AMPL significantly contributed to my work.

I am also thankful to my mentor, Jaime Godoy Calvo, for his practical advice and patience during my time in the lab. I appreciate the helpful assistance from Alejandro Salazar Gómez and Daniel Serrano Domínguez.

I would like to express my sincerest gratitude to Dr. Narcis Palomeras and Dr. Nuno Gracias for their academic guidance and support throughout the IFROS program. I am also thankful to Imma Güell for her kind support.

Finally, I want to express my deepest gratitude to my family and friends for their support and encouragement. Sizlerin desteği yolumu aydınlattı, sağ olun var olun.¹

Thank you.

¹Your support has illuminated my path, thank you, and may you always be well.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	iii
List of Tables	iv
Glossary and Acronyms	v
1 Introduction	1
1.1 Context	1
1.2 Problem Statement	2
1.3 Research Questions	3
1.4 Contribution	3
1.5 Outline	3
2 Background	5
2.1 Object Detection	5
2.2 Data Annotation	6
2.3 3D LiDAR Point Clouds	7
2.4 Projection to 3D LiDAR Point Clouds	8
2.5 Relevant Datasets	8
2.6 Performance Evaluation Metrics	10
3 Related Work	12
3.1 R-CNN Family	12
3.2 YOLO	15
3.3 Transformer-Based Models	18
3.4 Comparison of Object Detection Approaches	20

CONTENTS

3.5	2D-3D Projection	20
3.5.1	Frustum PointNets [12]	21
3.5.2	PointPainting [36]	21
3.6	Outlier Rejection and Geometric Fitting	22
3.6.1	RANSAC	22
3.6.2	MAGSAC	23
3.7	SAHI	23
4	Methodology	24
4.1	Overview of the Methodology	24
4.2	Data Collection and Preprocessing	25
4.2.1	2D Detection: Dataset Selection and Preprocessing	25
4.2.2	3D Detection: Sequence Selection and Preprocessing	26
4.3	Model Architecture and Training	27
4.3.1	2D Detection: Model Architecture and Training	27
4.3.2	3D Detection: Geometric Modeling and Inference	28
4.3.3	Simplifications and Assumptions	30
5	Results and Evaluation	32
5.1	Training Setup and Configuration	32
5.2	2D Detection Performance	32
5.2.1	SAHI Evaluation	33
5.3	3D Localization Results	34
5.3.1	Qualitative Evaluation	36
5.4	Performance Evaluation	36
6	Conclusion and Future Work	39
6.1	Summary of Contributions	39
6.2	Limitations	40
6.3	Future Work	41
	Bibliography	41
A	Annotation Formats	47

List of Figures

2.1	Comparison between concepts, retrieved from [8]	6
2.2	Comparison of Formats	7
2.3	3D Viewing Frustums, adapted from [12]	9
2.4	ATLAS Car of AMPL Lab [16]	10
3.1	R-CNN Pipeline from Girshick et al. [20]	13
3.3	Faster R-CNN by Ren et al. [9]	14
3.2	Fast R-CNN by Girshick [23]	14
3.4	YOLO Detection System from Redmon et al. [21].	16
3.5	YOLO Model from Redmon et al. [21].	16
3.6	Non-Maximum Suppression (NMS), adapted from [34]	18
3.7	DETR Model from Carion et al. [2]	19
3.8	Overview of RT-DETR Architecture [3]	19
3.9	Frustum PointNets Pipeline, retrieved from [12]	21
3.10	PointPainting Pipeline, retrieved from [36]	22
4.1	Overview of Methodology	25
5.1	Original Image and Detection	33
5.2	Detection with SAHI Enhancement	34
5.3	Projection of 2D Detection Box	35
5.4	Outlier Rejection	35
5.5	Sequence displaying the 3D bounding box construction in eight steps.	38

List of Tables

3.1	YOLO Evaluation, adapted from [31]	18
3.2	Qualitative comparison of prominent object detection models across key performance criteria. Based on findings from [3], [21], and [9].	20
5.1	2D detection results for the fine-tuned YOLO model (confidence $\geq$ 0.25).	33
5.2	Performance comparison of the fine-tuned model with and without SAHI.	34
5.3	3D Localization Results	36

Glossary and Acronyms

AP Average Precision. 23

CNN Convolutional Neural Network. 5

COCO Common Objects in Context. 2, 6, 7, 15–17, 26, 27

DETR Detection Transformer. 2, 18

JSON JavaScript Object Notation. 6, 26

LiDAR Light Detection and Ranging, a remote sensing method that uses laser light to measure distances. 1, 3, 7–9, 20, 21, 26, 28–31, 34, 39

MAGSAC Marginalizing Sample Consensus. 23

Non-Maximum Suppression A technique used in object detection to remove redundant overlapping bounding boxes by keeping only the one with the highest confidence score. 2

R-CNN Region-based Convolutional Neural Network. 4, 6, 12–14

RANSAC Random Sample Consensus. 22, 23

RGB Red Green Blue. 2, 21, 26

RT-DETR Real-Time DETection TRansformer. 2, 19, 20

SAHI Slicing Aided Hyper Inference. 23

SfM Structure from Motion. 2

TXT Plain Text File Format. 6

YOLO You Only Look Once. 2, 4, 6, 7, 15, 17, 18, 26, 27, 32

Chapter 1

Introduction

1.1 Context

Autonomous systems operating in urban environments face great challenges due to the complexity and variety of the cityscapes. Unlike controlled settings like warehouses and factories, urban areas contain diverse objects, like vehicles, pedestrians, and traffic signs, creating significant scene variability. Ensuring autonomous systems can detect and understand these elements reliably is critical for safe and intelligent navigation.

Deep learning has significantly advanced detection algorithms, with established approaches like Convolutional Neural Networks [1] and more recent Transformer-based detectors [2] improving performance in visual perception tasks. However, relying solely on camera images can be limiting, as the lack of spatial context poses challenges for ensuring safe and reliable navigation.

To address this limitation, the integration of multiple data sources—commonly referred to as Sensor Fusion—has emerged as a preferred approach. Sensor Fusion allows the system to benefit from the strengths of each sensor, like strong visual cues from the cameras and the precise spatial information from the LiDAR. This combination allows a more comprehensive and reliable understanding of the environment.

However, accurate detection and localization of small but critical urban objects, such as poles, remain challenging due to factors like occlusions, varying lighting conditions, and sensor noise. This thesis addresses these challenges by improving sensor data integration to enhance environmental perception and support safer navigation.

1.2 Problem Statement

An important problem in urban detection is to ensure the accuracy in real-world conditions. However, objects in urban settings are frequently occluded by other elements like pedestrians, trees, vehicles, etc, which makes it difficult for object detectors to work properly. Additionally, in urban environments, there is a high variability in lighting conditions, which affects the performance of vision-based detection models.

Despite recent advancements in deep learning-based object detection, the existing models still struggle with real-world conditions [2]. Transformer base models such as DETR [2] and RT-DETR [3] have shown improvements in handling occlusions and reducing dependency on post-processing techniques like Non-Maximum Suppression. However, their high computational cost and slower inference times make it difficult to use these models in real-time applications like autonomous driving or mobile robotics, where fast processing and efficient use of resources are essential. On the other hand models like YOLO (You Only Look Once) offer fast speed but may lack robustness in complex urban scenarios.

Another critical problem is the need for high-quality annotated data sets that accurately represent real-world urban environments and encompass the annotations for the required classes. While large-scale datasets such as COCO [4], KITTI [5], and NuScenes [6] provide valuable data for training object detection models, they lack the annotations required to detect pole-like objects.

Furthermore, the detections of the pole-like objects are typically gathered from monocular RGB images and therefore lack depth information. Without stereo vision, temporal motion, or prior scene knowledge, the location of the objects cannot be known reliably. While SfM [7] or monocular depth estimation using deep learning can be used to estimate the depth of the object, SfM requires sequential frames or video data, and monocular depth estimation is reliable when the trained dataset and the used environment are similar, and the weather conditions are stable. Overall, the limitations are significant.

1.3 Research Questions

The study investigates the effectiveness of combining 2D image-based data and 3D spatial information for urban detection, and addresses the following research questions:

1. How can urban objects be accurately detected using a combination of 3D data and image-based deep learning models?
2. What improvements can sensor fusion (e.g., LiDAR and cameras) bring to detection accuracy and robustness?
3. How can the developed system handle challenges such as occlusions, cluttered scenes, and dynamic environments effectively?

1.4 Contribution

This thesis contributes to the field of autonomous navigation by incorporating 2D deep learning-based object detection into a 3D spatial representation. The main contributions are listed as follows:

- A 2D Object Detector that is fine-tuned for pole-like structures using the YOLOv11 architecture. This targets a class that is generally overlooked in object detection datasets.
- A 3D localization pipeline that projects the 2D detection outputs onto LiDAR data, which allows the estimation of the spatial positioning of the pole objects.
- The approach of the thesis shows how combining segmentation-derived labels with geometric modelling can be used as a lighter alternative to 3D object inference.

This thesis enhances the localization of urban objects by utilizing both 2D and 3D data.

1.5 Outline

This thesis is organized into six chapters. Following the Introduction, Chapter 2 presents the essential background and the literature review, such as broader knowl-

1. Introduction

edge on object detection models, relevant datasets, data annotation techniques, performance evaluation metrics, and state-of-the-art methods such as the R-CNN family, the development of YOLO, and transformer-based models. Chapter 3 details the methodology detailing the main steps of the project like the data collection and preprocessing procedures, model architecture, training approaches, evaluation metrics, and implementation details. Chapter 4 presents the results and evaluation of the thesis work, analyzing the 2D detection performance, the 3D localization results, and discussing the findings. Finally, Chapter 5 concludes the thesis with the summary of the contributions, limitations of the study and suggestions for future work.

Chapter 2

Background

This chapter provides the necessary background for the conducted work. It outlines the fundamental concepts and domain-specific knowledge required to understand the study.

2.1 Object Detection

Object detection is a central task in many computer vision applications. It involves both the localization and the classification of the object in an image. Each detected object is classified into a predefined category, and its position is indicated in the image. Due to its combined classification and localization abilities, object detection is fundamental to applications in autonomous systems, healthcare diagnostics, and surveillance technologies.

Although image classification, localization, and object detection employ related technologies, their objectives and outcomes differ significantly. Classification model predicts a single class label for an image, meanwhile object detection a multi-instance recognition by locating all the relevant objects. Although object detection is commonly used with multi-class detection tasks, it can also be configured for single-class detection, focusing on detecting the instances of a specific object category. Figure 2.1 displays the conceptual difference of these tasks.

Traditional object detection techniques often relied on computationally expensive sliding window approaches. Early Convolutional Neural Networks (CNNs) also employed sliding windows for object detection, but the development of more efficient architectures has greatly advanced the field. CNN-based models can learn hierarchi-

2. Background

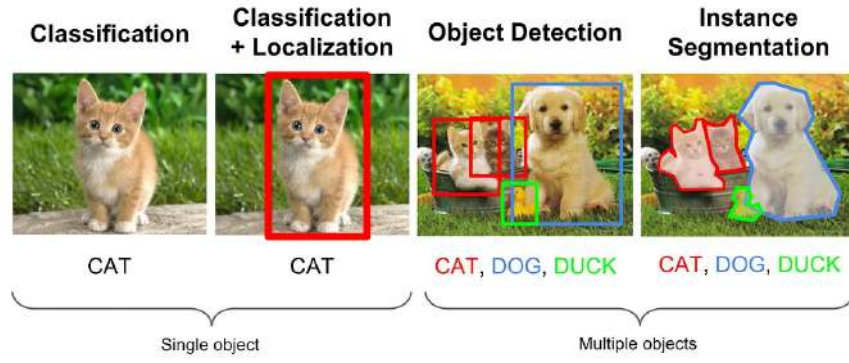


Figure 2.1: Comparison between concepts, retrieved from [8]

cal feature representations directly from data, resulting in substantial improvements in detection accuracy and speed [1]. Notable models such as Faster R-CNN [9] and YOLO have significantly enhanced the performance of object detection systems.

2.2 Data Annotation

There are different annotation strategies that can be used in computer vision tasks. The most basic one of these is the bounding box format. Bounding boxes define rectangular coordinates to indicate the location of the object. While bounding boxes are very efficient, the precision can be limited if the mentioned object is not rectangular in shape. In that case, the bounding box rectangle may encompass parts of other object classes than the one being annotated. For those objects, semantic segmentation can be a better choice, especially for multiple object detection, as semantic segmentation has pixel-level accuracy and masks the objects inside the image, albeit with a higher cost. Although semantic segmentation does not give unique labels to each instance of an object, and to achieve distinct labelling for each object, instance segmentation can be used, according to [10].

Other annotation formats include polygon annotations, which approximate the object shape through connected vertex points, and 3D cuboids (also known as 3D bounding boxes), which represent the object with depth information.

Several standardized annotation frameworks have been adopted in the computer vision community. The COCO format uses the JSON structure that includes the annotations of all the images in a dataset. YOLO employs a different format that consists of a set of TXT documents for each image, containing as many lines as

2. Background

the annotated objects in that image. Other established formats include Pascal VOC with its XML-based structure.

In addition to file structures being different in COCO and YOLO format, they also have differences in the calculation of the object location. While COCO format includes the pixel location (x,y) of the top-left corner of the bounding box, and the bounding box width and height, YOLO format uses a much more normalized calculation. YOLO format includes the pixel location (x,y) that is normalized by the image width and height, and the bounding box width and height that is normalized by the image width and height. Figure 2.2 shows the comparison of formats. Formatting rules for the different file types can be found in Appendix ??.

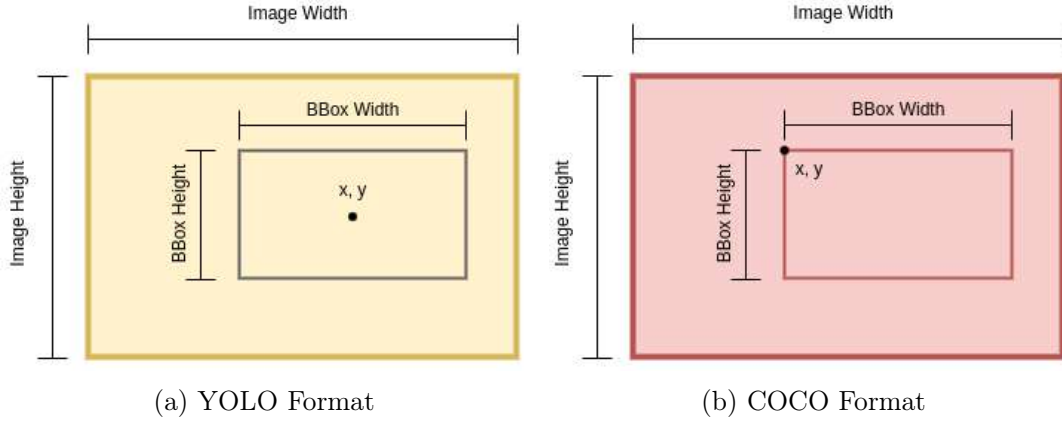


Figure 2.2: Comparison of Formats

2.3 3D LiDAR Point Clouds

Light Detection and Ranging (LiDAR) is a sensing technology measuring distances by emitting laser pulses and capturing their reflections. The resulting data are structured as three-dimensional point clouds, where each point is defined by spatial coordinates (x,y,z) and in some cases additional attribute like intensity values. These point clouds provide high-resolution spatial information that captures the structure of the surrounding environment.

3D LiDAR point clouds serve as an important source of spatial data in autonomous navigation, robotics, and environmental mapping. They are crucial in modeling of complex scenes and they support perception tasks, including localization, mapping, and object detection. In contrast to image data, point clouds

2. Background

are irregular, sparse, and lack a consistent grid structure, which introduces specific challenges for data representation and processing.

2.4 Projection to 3D LiDAR Point Clouds

Projection to 3D LiDAR point clouds using the 2D detections is a crucial step to understand the scene comprehensively. Projection is performed based on the pin-hole camera model, which mathematically defines the relationship between 2D image coordinates and 3D world coordinates through intrinsic and extrinsic calibration parameters [11]. Equation 2.1 illustrates the pinhole camera projection model. The vector $\begin{bmatrix} u & v & 1 \end{bmatrix}^T$ denotes the homogeneous image coordinates, $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ represents the 3D world coordinates, and $\mathbf{K}$ refers to the camera intrinsic matrix. The intrinsic matrix $\mathbf{K}$ represents the internal parameters of the camera, where f_x and f_y are the focal lengths in pixels along the x- and y-axes, respectively, and (c_x, c_y) are the principal point coordinates. The term $\begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix}$ corresponds to the camera extrinsic parameters, consisting of the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \text{where} \quad \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.1)$$

Using this projection equation in autonomous vehicles requires precise LiDAR sensor and camera calibration. Well-calibrated parameters enable the system to project LiDAR point clouds onto the image plane and accurately transform LiDAR point clouds into the coordinate system of the camera. A common approach for associating 2D detections with 3D LiDAR points involves creating 3D viewing frustums from 2D bounding boxes [12]. These frustums that are shaped like pyramids effectively filters the point cloud to isolate the points that potentially belong to the object to be detected [12], as illustrated in Figure 2.3.

2.5 Relevant Datasets

Dataset selection plays a critical role in computer vision applications. It is important to pick a dataset that aligns with the specific requirements of the project.

2. Background

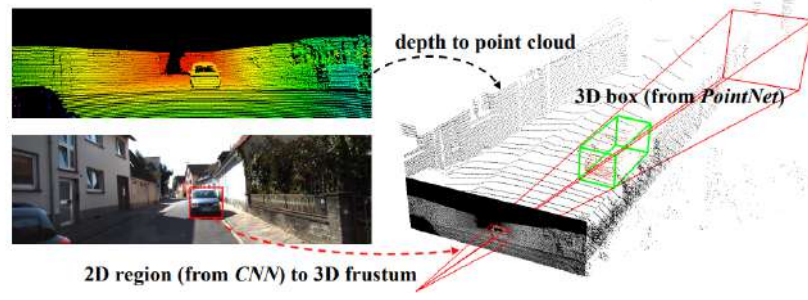


Figure 2.3: 3D Viewing Frustums, adapted from [12]

The datasets considered for this study were evaluated based on modality, annotation quality, annotated object types, and diversity of urban scenes. Given the nature of the problem, the dataset selection process was categorized into two groups for the 2D and 3D Parts.

2D Dataset. For the 2D component of the project, datasets with comprehensive annotations of urban objects were necessary. The following datasets were considered:

- **NuScenes [6]** Bounding box annotations for traffic signs, barriers, and poles with high-quality 32-beam LiDAR and six-camera coverage in urban settings.
- **Argoverse 2 [13]** Features high-definition maps and lane information, allowing a comprehensive study of object detection in structured urban environments.
- **A2D2 [14]** Offers dense and diverse annotation for urban objects, including poles.
- **Zenseact Open Dataset (ZOD) [15]** Includes annotations for street poles, barriers, and infrastructure, with data collected under adverse weather conditions, useful for robustness analysis.

3D Dataset. The sequence used in 3D Phase is recorded by ATLAS the car (shown in Figure 2.4, and encompasses Basler acA2040-35gc 4mm focal length cameras and Velodyne VLP-32C LiDAR.

2. Background



Figure 2.4: ATLAS Car of AMPL Lab [16]

2.6 Performance Evaluation Metrics

Several metrics are commonly employed to quantitatively assess the performance of object detection models during evaluation. The most common metrics for evaluating detection accuracy are Intersection over Union (IoU) and mean Average Precision (mAP) [17]. To understand these metrics, it is essential to be familiar with true positives, false positives, and false negatives.

Intersection over Union (IoU). Intersection over Union, also known as the Jaccard Index, is the most commonly used metric for comparing the similarity of two shapes [18].

- **True Positives (TP)** True Positives are correct detections of the ground truth of a bounding box
- **False Positives (FP)** False Positives are the detection of an object that does not exist, or a wrongly-placed detection of an existing object
- **False Negatives (FN)** False Negatives are undetected bounding boxes

$$\text{IoU} = \frac{|P \cap G|}{|P \cup G|} = \frac{|P \cap G|}{|P| + |G| - |P \cap G|} \quad (2.2)$$

where P represents the set of points in the predicted object region, G represents the set of points in the ground truth object region, and $|\cdot|$ denotes the cardinality, corresponding to the area or number of elements in the set.

2. Background

In the context of object detection, for the detection to be considered correct, the IoU should be higher than the decided threshold, which is typically chosen based on the characteristics of the object.

mean Average Precision (mAP). Knowing what is Precision and Recall is preliminary to understand what is mean Average Precision.

Precision is the ability of the model to identify the detections over all detections correctly. It captures how focused the model is in its predictions, only labeling an object when it's confident it's truly there. Meanwhile, Recall is the ability of the model to identify all the relevant cases.

$$\text{Precision} = P = \frac{TP}{TP + FP} \quad (2.3)$$

$$\text{Recall} = R = \frac{TP}{TP + FN} \quad (2.4)$$

Precision is more important when False Positives result in a higher cost, like detecting important emails as spam. On the other hand, Recall is more important in scenarios where missing a relevant case is costly, like diagnosing critical diseases.

The Precision-Recall curve plots precision values against corresponding recall values across different confidence thresholds. It forms a curve because precision and recall display a trade-off. Adjusting the confidence threshold results in different pairs of precision and recall values, hence reflecting the balance between false positives and false negatives. The area under this curve is called Average Precision (AP) or Area Under Curve (AUC), which provides a single metric that summarizes the model's performance. Mean Average Precision is calculated as the average of the AP over all classes to be detected.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N AP_i \quad (2.5)$$

Chapter 3

Related Work

Object detection methods have transformed and improved remarkably over the years. While the earlier object detection models relied on traditional approaches like sliding windows and handcrafted features proposed in [19], modern object detection models use deep learning architectures that automatically learn hierarchical feature representations, significantly improving efficiency and accuracy.

Modern object detection models can be discussed in three categories: two-stage detectors that generate region proposals before classification [20], single-stage detectors that predict the object locations and classes simultaneously [21], and the transformer-based approaches that reformulate object detection as a set prediction problem using attention mechanisms to directly model object relationships and global context [2].

These methodologies present trade-offs between detection accuracy, computational efficiency, and architectural complexity.

3.1 R-CNN Family

Originally suggested by [20], region-based convolutional networks presented a significant change in object detection by decomposing the process into region proposals and classification. The R-CNN architecture Girshich et al. [20] builds on the earlier work on selective search by Uijling [22], extracting some regions from the images as region proposals, and decreasing the computational search space, shown in Algorithm 1. This way, computational resources can be used to identify promising image regions.

3. Related Work

Algorithm 1 Selective Search by [22]

- 1: **Input:** Image I
 - 2: **Output:** Region proposals $\mathcal{R}$
 - 3: Generate initial sub-segmentation of I to obtain small regions
 - 4: **while** there are similar regions **do**
 - 5: Identify the most similar pair of regions
 - 6: Merge them into a larger region
 - 7: **end while**
 - 8: Return all merged regions as candidate proposals $\mathcal{R}$
-

The pipeline of R-CNN, also shown in Figure 3.1, extracts around 2000 region proposals per image using the selective search algorithm, computes CNN features for each, and classifies them using SVMs. This method has provided 30% relative improvement to the best previous result on PASCAL VOC 2012. However, R-CNN had essential limitations, such as slow inference due to multi-stage training and redundant feature computation.

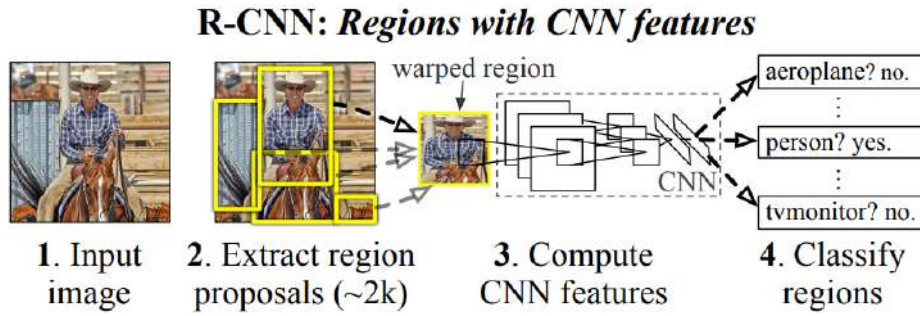


Figure 3.1: R-CNN Pipeline from Girshick et al. [20]

Fast R-CNN. Fast R-CNN [23] addressed some of the limitations of R-CNN. First, feature extraction is not done for each region, but only for the image. Then, a Region of Interest (RoI) pooling layer is applied to extract fixed-size features from the feature map. Finally, these features are passed through the fully connected layers, which branch into a softmax classifier and a bounding box regressor. Figure 3.2 shows the pipeline of the Fast R-CNN architecture.

3. Related Work

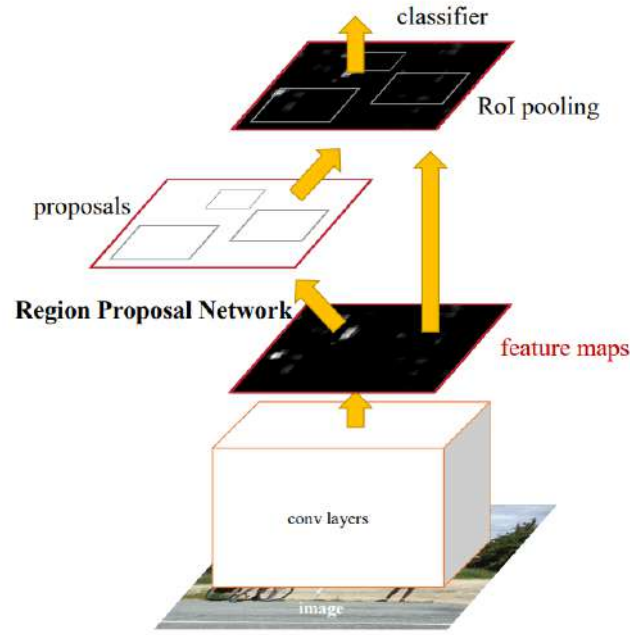


Figure 3.3: Faster R-CNN by Ren et al. [9]

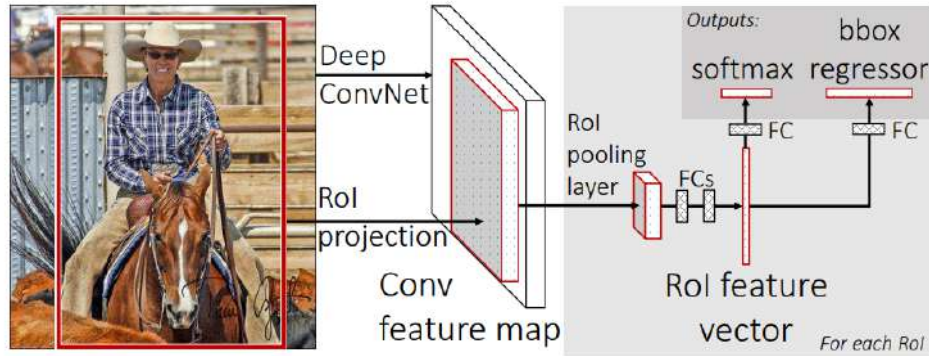


Figure 3.2: Fast R-CNN by Girshick [23]

Faster R-CNN. Faster R-CNN by Ren et al. [9] completed the evolution of the R-CNN family: a fully end-to-end trainable detector by replacing Selective Search with a Region Proposal Network (RPN). Like Fast R-CNN, the image is input into a CNN that provides the feature map; however, instead of using the Selective Search algorithm, RPN is used. RPN introduces anchor boxes at different scales and aspect ratios. This integration eliminates the need for an external proposal generation method while improving proposal quality through task-specific learning. Figure 3.3 shows the pipeline of the Fast R-CNN architecture.

3.2 YOLO

YOLO [21] is an important model widely used for object detection. It effectively addresses real-time object detection challenges by balancing speed and accuracy. Unlike sliding windows or region-based proposals, YOLO scans the entire image to understand contextual information [21]. Since its foundation in 2016, YOLO has undergone many changes and improvements [24].

YOLOv1. The core principle of the original YOLO, YOLOv1, involves imposing a grid cell with $s \times s$ dimensions onto an image, detecting the object's center if it falls into one cell, allowing others to ignore it [21]. This approach achieved very high inference speed (45 FPS on Titan X GPU) while maintaining a competitive accuracy (63.4% mAP on PASCAL VOC, 21% mAP on COCO), which is suitable for real-time applications [21]. However, the first version of YOLO had limitations in small object detection and precision in localization. YOLO detection system and model is shown in Figure 3.4 and Figure 3.5 respectively.

3. Related Work

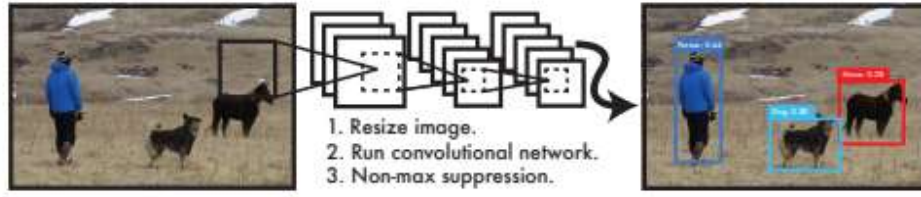


Figure 3.4: YOLO Detection System from Redmon et al. [21].

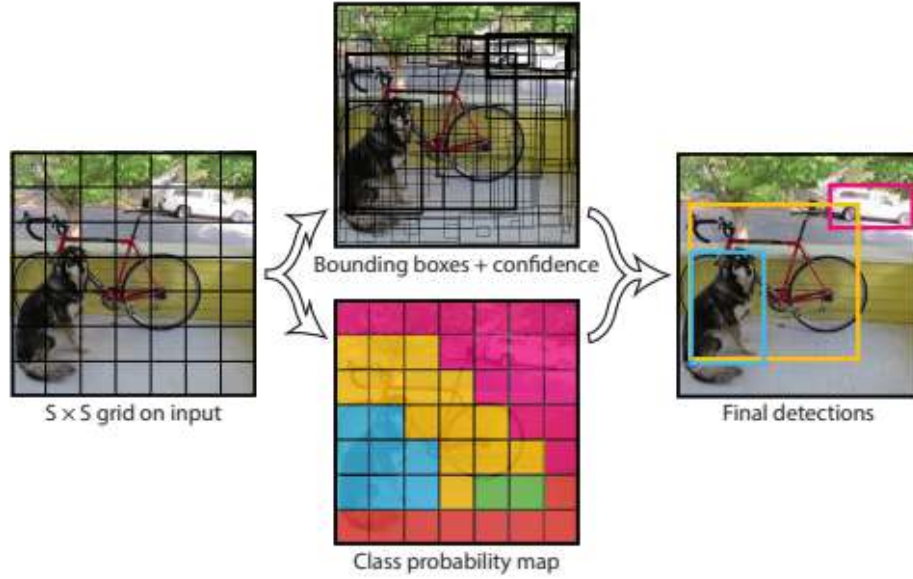


Figure 3.5: YOLO Model from Redmon et al. [21].

YOLOv3. YOLOv3 [25] marked an important advancement by using multi-scale prediction approach and Darknet-53 deeper feature extractor. YOLOv3 significantly improved the limitations of YOLOv1 by improving small object detection thanks to prediction in different scales and utilizing a feature pyramid network. It achieved 28.2% COCO mAP at 45 FPS (320x320) and 33.0% COCO mAP at 19.6 FPS (608x608) [25].

YOLOv4. Bochkovskiy et al. [26] incorporated some architectural improvements, including PANet for parameter aggregation instead of FPN that was used in YOLOv3, CSPDarknet53 as its backbone instead of Darknet53 in YOLOv3, and mosaic data augmentation [26][25]. These improvements resulted in 43.5% COCO mAP (IoU 0.50:0.95) at 65 FPS on Tesla V100 GPU [26].

3. Related Work

YOLOv5. Ultralytics' YOLOv5, the first YOLO version introduced in PyTorch, offers scale model variants to solve some of the computational constraints [27]. The model YOLOv5L (640 px) achieved 49.0% COCO mAP (IoU 0.50:0.95) at around 99 FPS on V100.

YOLOv8. YOLOv8 by Ultralytics, introduces an anchor-free detection head with C2f block improvements[28]. It further improves late-stage fine-tuning by stopping mosaic augmentation in the final epochs [28]. These changes introduce around %3 improvement in mAP (COCO) at similar versions like YOLOv5l and YOLOv8l. YOLOv8 also offers a simple API to connect the model with several downstream tasks like instance segmentation, pose estimation, etc [29].

YOLOv11. The current state-of-the-art YOLOv11 [30] that is also used in this research represents the culmination of the YOLO evolution. a significant improvement is the introduction of C3k2 block that is more computationally efficient than the C2f block that was used in YOLOv8 [31]. It also introduces a new Cross Stage Partial wit Spatial Attention (C2PSA), which increases the spatial attention mechanism [31]. With its key improvements, YOLOv11 (YOLOv11m) reduces %22 of the parameters compared to YOLOv8m with an increase in mAP on COCO [30]. The YOLOv11x achieves %53.5 mAP(50-95) at 13m inference speed, which over-performs all the previous YOLO iterations [31]. Its performance in low-latency is also remarkable, YOLOv11s achieves %47 mAP(50-95) while the earlier low-latency versions are much less accurate [31]. Additionally, it is very versatile in computer vision tasks as it can be used for instance segmentation, image classification, pose estimation, and oriented object detection [32].

While there has been remarkable advancements throughout the YOLO evolution, some challenges stay inherent in the YOLO architecture. One such challenge is the reliance on Non-Maximum Suppression (NMS), illustrated in Figure 3.6, which is necessary for eliminating redundant detections but can limit both detection speed and accuracy [3]. Nevertheless, YOLO models successfully adapted to many industrial and academic projects due to their accessible implementations and extensive community support. The widespread adoption of the model allowed many developers to benefit from the well-documented repositories and resources during the

3. Related Work

Model	Year	Contributions	Framework
YOLO [21]	2015	Single-stage object detector	Darknet
YOLOv2 [33]	2016	Multi-scale training; dimension clustering	Darknet
YOLOv3 [25]	2018	SPP block; Darknet-53 backbone	Darknet
YOLOv4 [26]	2020	Mish activation; CSPDarknet-53 backbone	Darknet
YOLOv5 [27]	2020	Anchor-free detection; SWISH activation; PAnet	PyTorch
YOLOv8 [28]	2023	GANs; anchor-free detection	PyTorch

Table 3.1: YOLO Evaluation, adapted from [31]

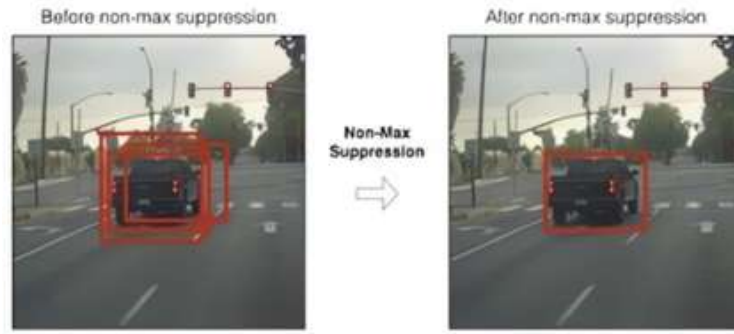


Figure 3.6: Non-Maximum Suppression (NMS), adapted from [34]

implementation process. An overall comparison between the YOLO versions and their comparisons is displayed in Table 3.1.

3.3 Transformer-Based Models

Recent developments have led to the creation of end-to-end object detectors based on transformers. Carion *et al.* [2] first proposed DETR, an end-to-end detector based on transformers, which has gained significant attention in academia as it eliminates the need for hand-crafted anchors and NMS components. However, DETR is unsuitable for real-time detection due to its high computational cost, slow training convergence, and the need for large training data [3].

3. Related Work

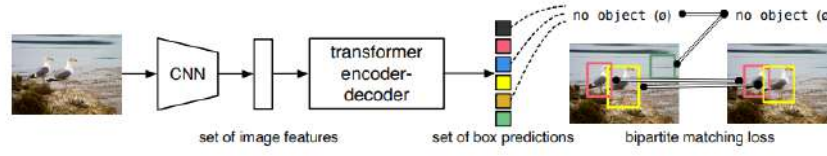


Figure 3.7: DETR Model from Carion et al. [2]

To address these challenges, RT-DETR (Real-Time Detection Transformers) was introduced by Zhao *et al.* [3]. RT-DETR builds upon DETR’s transformer backbone and introduces key modifications to make it suitable for diverse and real-world scenarios.

The core improvement of RT-DETR is the integration of mechanisms that allow the model to handle better noisy data, varying object sizes, and occlusions.

Some key features of RT-DETR, as discussed by Nandini Lokesh Reddy in [35], include:

- **Hybrid Encoder:** RT-DETR uses a hybrid encoder instead of a vanilla Transformer encoder to efficiently process multi-scale features, which significantly improves inference speed.
- **Uncertainty-Minimal Query Selection:** RT-DETR uses an Uncertainty-Minimal Query Selection Scheme that explicitly constructs and improves epistemic uncertainty [3]. This method selects the best initial queries for the decoder, optimizing accuracy [35].

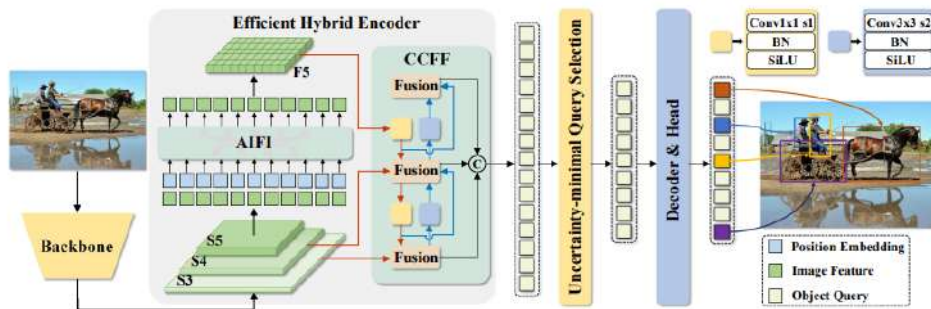


Figure 3.8: Overview of RT-DETR Architecture [3]

An overview of the RT-DETR architecture can be seen in Figure 3.8. The features are fed into the encoder from the last three stages of the backbone. The hybrid encoder converts multi-scale features into an image feature sequence by utilizing

3. Related Work

Attention-based intra-scale feature interaction (AIFI) and CNN-based Cross-scale Feature Fusion (CCFF). Later, the uncertainty-minimal query selection chooses a fixed number of encoder features to initialize the object queries. Finally, the decoder refines the object queries to produce the labels and bounding boxes [3].

However, RT-DETR models are quite large and difficult to train compared to other suggested models. For single-object detection tasks, using such a large model could be unnecessarily complex and lead to issues associated with a bigger model size.

3.4 Comparison of Object Detection Approaches

Model	Inference Speed	Detection Accuracy	Robustness to Occlusion	Real-Time Suitability	Training Time
Faster R-CNN	Slow	Very High	Moderate	Low	Long
YOLO	Very Fast	Good	Moderate	Excellent	Short
RT-DETR	Moderate	High	High	Moderate	Long

Table 3.2: Qualitative comparison of prominent object detection models across key performance criteria. Based on findings from [3], [21], and [9].

Object detection models present distinct trade-offs when evaluated for accuracy, speed, and computational complexity. While Faster R-CNN and RT-DETR offer higher accuracies, YOLO’s much shorter training time and unmatched speed make it the most suitable detection model for this project. Table 3.2 displays the comparison between the models.

3.5 2D-3D Projection

Integrating information from camera-based detections and geometric data from LiDAR requires projection methodologies that can effectively bridge the dimensional gap between 2D image observation and 3D spatial understanding. This integration presents challenges in maintaining spatial accuracy across different sensor modalities.

3. Related Work

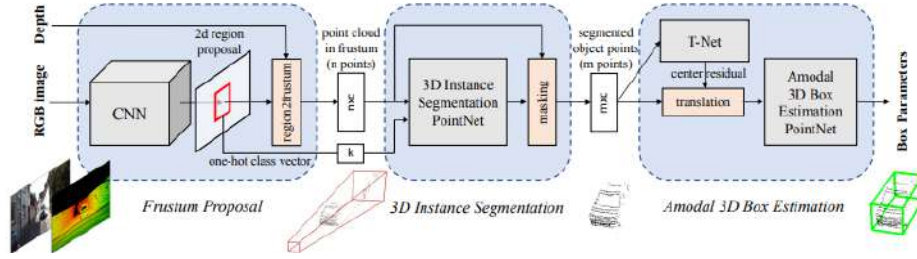


Figure 3.9: Frustum PointNets Pipeline, retrieved from [12]

3.5.1 Frustum PointNets [12]

Frustum PointNets [12] addressed the 3D object detection problem through a sequential decomposition strategy leveraging 2D detection constraints. The methodology starts with bounding boxes retrieved by a 2D object detector. Each 2D bounding box undergoes back-projection into 3D space through calibration parameters, thereby defining a frustum volume.

LiDAR points within the defined frustum are extracted and processed as an independent subset. The approach employs a PointNet-based segmentation network to identify the points within the subset that belong to the target object. In addition, Frustum PointNets [12] incorporates a transformation network (T-Net) to address spatial alignment problems by predicting residual center corrections. Subsequently, a regression network estimates the 3D bounding box parameters.

While achieving computational efficiency, Frustum PointNets [12] inherits dependencies from the initial 2D detector accuracy and the precision of the geometric projection process. These dependencies become particularly important when dealing with occluded objects. The Frustum PointNets pipeline is displayed in Figure 3.9.

3.5.2 PointPainting [36]

PointPainting [36] presents an alternative integration strategy that directly implants semantic information into the point cloud representation. The approach begins with a semantic segmentation network that processes RGB images to generate pixel-wise probability distributions among classes. LiDAR points are then mapped onto the image plane, where the matching pixel positions give semantic probability values.

3. Related Work

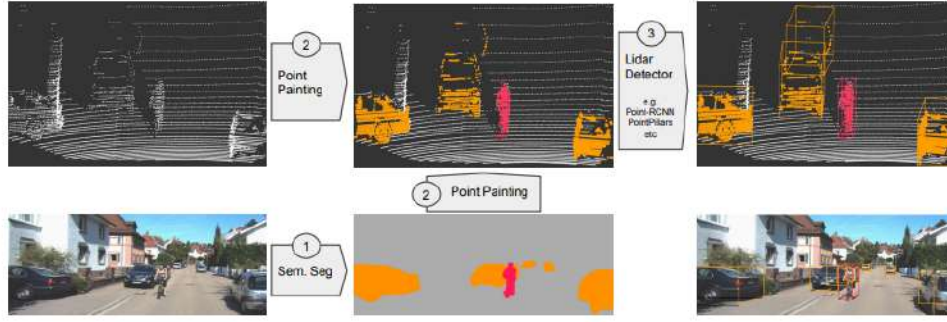


Figure 3.10: PointPainting Pipeline, retrieved from [36]

This augmentation process results in point representations that combine geometric spatial information with semantic context. The semantic-enhanced point clouds are fed into a 3D Detector, such as PointPillars [37]. The PointPainting pipeline is shown in Figure 3.10.

3.6 Outlier Rejection and Geometric Fitting

Subsequent to projection processes, geometric model fitting becomes vital for determining object structures. Therefore, robust algorithms are required to accurately process sensor data to address problems stemming from noise and outliers.

3.6.1 RANSAC

Random Sample Consensus (RANSAC) [38] is an iterative model estimation framework through minimal subset sampling. The algorithm works by repeatedly selecting the minimum number of data points required to instantiate the model hypothesis, such as lines, planes, and circles. It assesses the quality of each hypothesis by determining the number of inliers, which are the points that lie within a specified distance tolerance. The model receiving the highest amount of inliers across all iterations is selected as the final estimate.

RANSAC [38] demonstrates effectiveness with substantial outlier presence, though its performance depends on the calibration of the parameters, such as the iteration count, inlier threshold and minimal subset size.

3.6.2 MAGSAC

Marginalizing Sample Consensus (MAGSAC) addresses the limitations of RANSAC through a probabilistic model marginalization approach. Rather than relying entirely on inlier counts, MAGSAC evaluates the model hypothesis with scoring functions incorporating point distance likelihoods under assumed noise models. This framework eliminates the need for inlier thresholds and provides more robust estimation in ambiguous geometric configurations. The method computes soft inlier scores based on predicted patterns of measurement errors, and subsequently aggregates these scores across iterations to identify the best fit.

3.7 SAHI

The Slicing Aided Hyper Inference (SAHI) framework represents a significant advancement in small object detection methodologies[39]. One of the main problems with small object detectors is that such objects often occupy only a few pixels in high-resolution images, making them easily overlooked by standard detection models. SAHI implements a slicing-based approach that divides the image into overlapping sections, increasing the coverage of small objects that might otherwise be missed in full-image detection. Each sliced section is passed through the object detector independently, and the resulting predictions are then aggregated and reconciled to form the final output. This method does not require changes to the architecture of existing object detectors and can be integrated on the model without re-training. It is compatible with widely used frameworks such as YOLOv5 [27], and experiments on aerial datasets including VisDrone [40] and xView [41] report improvements of up to 6.8% in AP when used at inference time. These results suggest that SAHI is a flexible and effective strategy for enhancing small object detection.

Chapter 4

Methodology

This section presents the methodology for urban object detection in autonomous navigation systems. The methodology consists of two main stages: (1) 2D object detection, which involves training a deep learning model to detect pole-like structures in urban environments, and (2) 3D object detection, where the 2D detections are projected into 3D space using sensor fusion and geometric modeling techniques. The section covers data collection and preprocessing, model architecture and training (for 2D), geometric modeling (for 3D), implementation details, simplifications, and assumptions.

4.1 Overview of the Methodology

The proposed methodology integrates 2D and 3D object detection approaches to create a comprehensive urban object detection system. As illustrated in Figure 4.1, the workflow begins with a rich dataset of urban images that undergoes preprocessing before being used for fine-tuning a pre-trained model. This 2D component produces a model that can detect pole-like objects in urban scenes. The trained model is subsequently employed to detect objects within the evaluation dataset. These 2D detections are then transformed into the 3D domain through projection, followed by point cloud clustering and outlier rejection processes to generate accurate 3D bounding boxes. This dual-stage approach utilizes the complementary strengths of both image-based deep learning techniques and geometric point cloud data to achieve robust detection and localization of urban objects.

4. Methodology

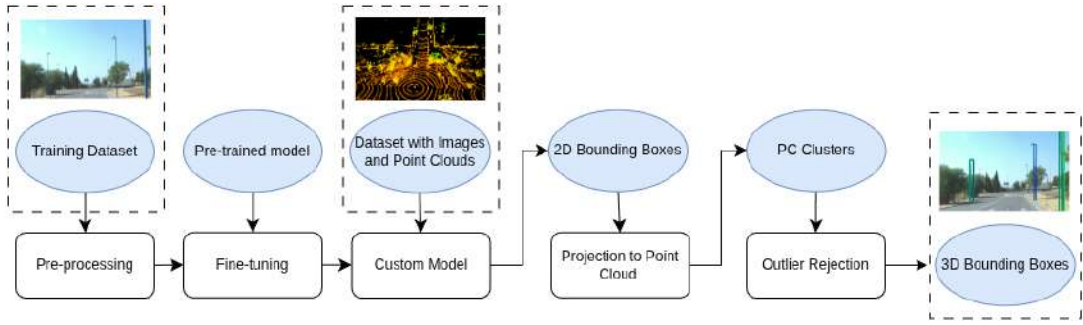


Figure 4.1: Overview of Methodology. Pre-processed training dataset fine-tunes a pretrained model. This model generates 2D bounding boxes, which are projected to 3D space using point clouds, followed by clustering and outlier rejection to produce final 3D bounding boxes

4.2 Data Collection and Preprocessing

4.2.1 2D Detection: Dataset Selection and Preprocessing

The first step in training the model for urban object detection was to focus on single-object detection, specifically targeting pole objects. Poles serve as structural supports for traffic signs, lampposts, traffic lights, and other urban infrastructure elements. Selecting a dataset with rich annotations for poles was crucial for this task.

Several datasets, including nuScenes and KITTI, were considered. However, these datasets lacked sufficient 2D bounding box annotations for poles, making them unsuitable for training a robust detection model. Given these limitations, the A2D2 dataset was selected. Although it lacks predefined 2D bounding box annotations, its accurate semantic segmentation labels made it a suitable choice, as they could be preprocessed and converted into a 2D bounding box format.¹

Preprocessing Details. A systematic preprocessing pipeline is proposed to convert the A2D2 Semantic Segmentation labels into 2D bounding boxes. This pipeline included loading the semantic images, applying image processing techniques, and then generating bounding boxes around these instances based on the segmented regions.

Bounding box annotations were generated first by loading each image, and consequently color thresholding was applied using predefined HEX color values to create

¹The laboratory where this master's thesis was conducted had prior experience working with A2D2, making it a practical choice for this project.

4. Methodology

masks, from which the contours are retrieved. Contours were then approximated to reduce complexity, and bounding box coordinates were determined using OpenCV's `boundingRect` function. To ensure high-quality detections, bounding boxes were filtered based on minimum height (130 pixels), maximum width (80 pixels), and minimum area (100 pixels). Threshold values were set to eliminate small objects that are unlikely to be poles (additionally, unlikely to be recognized by the detection model) and to ensure geometric consistency with the typical visual footprint of pole-like structures in urban settings. The processed bounding box information was stored in JSON format for further use. Parallel processing with multiprocessing was employed to accelerate annotation generation across large datasets.

Conversion of different formats. Bounding box outputs from different detection models were converted to a unified format to ensure compatibility with the project's processing pipeline. Conversion routines were implemented for both YOLO and COCO formats.

Splitting Dataset. Due to the fluctuating number of annotated objects throughout the sequences, a custom data splitting strategy was applied. The split was performed based on the cumulative number of annotated objects rather than the sequence count alone, and the split was made into training (80%), validation (10%) and test (10%) sets.

Once the splits were determined, only images that had at least one annotation were kept. Additionally, this process excluded unannotated images and handled missing files with appropriate warnings, ensuring that the final datasets were clean and consistent for the subsequent analyses. Overall, this preprocessing pipeline enabled the transformation of high-resolution segmentation masks into clean, standardized bounding box annotations in YOLO and COCO formats, suitable for model training.

4.2.2 3D Detection: Sequence Selection and Preprocessing

Unlike the 2D detection stage, which relies on publicly available, large, and annotated datasets for training and evaluation, the 3D detection stage is not designed for training purposes, but rather for inference and geometric processing. In this context, a short custom sequence consisting of RGB images and corresponding LiDAR

point clouds was used to demonstrate the application of the 3D projection method within the project environment.

4.3 Model Architecture and Training

This section describes the architecture and training process for the 2D object detection model, as well as the geometric modeling techniques employed in the 3D detection stage. While the 2D component involves supervised learning using a deep neural network, the 3D detection relies on inference and post-processing techniques to integrate depth information and extract spatial insights.

4.3.1 2D Detection: Model Architecture and Training

Model Selection and Overview. The object detection task in the 2D domain was addressed using the YOLOv11l architecture, which is the latest version of the YOLO family. YOLOv11l introduces architectural improvements that lead to enhanced multi-scale feature extraction and superior localization capabilities, which is important for a task that involves small and slender objects such as poles. YOLOv11l variant was chosen over the smaller n and s variants to maximize accuracy.

To maximize the performance of the model, YOLOv11l was initialized with the predetermined weights from COCO dataset². This initialization allows the model to leverage a broad understanding of common objects before fine-tuning it on the custom dataset. The custom dataset is derived from the A2D2 semantic segmentation data, and it is specifically adapted for pole detection in urban environments.

While it is possible to use YOLOv11 directly without fine-tuning, the COCO dataset does not contain specific annotations for poles. Instead, it includes annotations for similar but insufficient object categories, such as traffic signs or street furniture. Therefore, fine-tuning the model on a custom dataset allows for precise detection of poles, ensuring the model is tailored to the specific objectives of this project.

Parameters, Resource Constraints and Experimental Limitations. In this study, the model was trained using Ultralytics' standard YOLOv11L configuration, which includes default settings for optimization parameters such as learning rate,

²default for Ultralytics

4. Methodology

momentum, and data augmentation. These configurations are extensively validated by the framework and provided a robust baseline for training a single-class object detector on a custom dataset. Furthermore, leaving parameters untouched ensured reproducibility and helped to isolate the effects of dataset quality and annotation strategy on model performance.

Although additional improvements could be achieved through hyperparameter optimization or experimentation with different models, the scope of the experimentation was limited due to restricted access to GPU resources in the laboratory and the time-intensive nature of the training process. As a result, the configuration used in this study should be taken as a reliable starting point, rather than a fully optimized solution.

Additional Improvements. As an improvement, the Slicing Aided Hyper Inference (SAHI) method by Akyon et al. [39] could be applied to the model to enhance detection performance, particularly for this project where the target objects are thin and relatively small in size. SAHI [39] works by slicing the input image into overlapping patches, applying detection on each patch and then combine the results. While this approach can significantly improve precision and recall in complex scenes, it comes with the trade-off of slowing the process, making it less suitable for real-time applications. Nevertheless, it was included in this project as a potential enhancement.

4.3.2 3D Detection: Geometric Modeling and Inference

The system integrates 2D bounding boxes and LiDAR point clouds through projection-based sensor fusion. The core algorithm uses transformation of the 3D LiDAR points to the camera frame using calibrated extrinsic parameters and projects these points onto the 2D image. This process uses the detection results to focus on the analysis of the points.

Point Cloud Projection and Filtering. For each LiDAR point, the following transformation pipeline is applied:

- **Preliminary filtering.** LiDAR points that are behind the camera frame are eliminated.

4. Methodology

- **Frame Change.** LiDAR points are initially in the LiDAR frame. At this stage, the points are transformed to the camera coordinate system using the extrinsic calibration matrix.

$$\mathbf{p}_{\text{camera}} = \mathbf{T}_{\text{LiDAR} \rightarrow \text{camera}} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

- **Projection.** Later, the points in the camera frame are projected onto the image plane using the camera intrinsic matrix

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

- **Point Association.** Firstly, only the points within the image boundaries are considered. For each 2D detection box, the points that fall within its boundaries are associated with that detection, and the color information is transferred to the corresponding points.

Outlier Rejection with RANSAC. RANSAC-based line fitting is applied to the set of points associated with a detection box.

- **RANSAC Implementation.** The PCL's (Point Cloud Library) `SACSegmentation` module for RANSAC line fitting. The segmentation was configured to detect line models by setting the model type to `SACMODEL_LINE` and using `SAC_RANSAC` as the fitting method. A distance threshold of 0.02 meters was used to classify inliers during model estimation. Additionally, coefficient optimization was enabled to improve the accuracy of the fitted line model. The default PCL value of 50 iterations was used.
- **Inliers.** Points that within the line model from the above mentioned RANSAC algorithm are considered as inliers, and the non-inlier points are considered as outliers.

3D Bounding Box Estimation. The system generates the 3D bounding boxes using both the 2D detections and the 3D depth information.

4. Methodology

- **Depth Estimation and Inverse Projection.** Since the 3D bounding boxes are estimated in space, the corresponding 2D detection boxes must be reprojected back into 3D coordinates—effectively the inverse of the earlier projection step. For this purpose, the mean depth of the associated LiDAR points is computed and taken as the depth value Z_{center} for the bounding box. Then, each corner of the 2D bounding box is projected into 3D space using the inverse of the camera intrinsic matrix:

$$\mathbf{P}_{\text{corner}} = \mathbf{K}^{-1} \cdot \begin{bmatrix} x_{\text{corner}} \cdot Z_{\text{center}} \\ y_{\text{corner}} \cdot Z_{\text{center}} \\ Z_{\text{center}} \end{bmatrix}$$

where $\mathbf{K}^{-1}$ is the inverse of the intrinsic matrix and $(x_{\text{corner}}, y_{\text{corner}})$ is the 2D pixel coordinate of the bounding box corner.

- **Cylindrical Approximation.** Poles are approximated as cylindrical objects with circular cross sections. The width of the pole is calculated as the difference between the minimum and maximum depth values of the inlier points. This is used as both the width and the depth of the 3D box.
- **Bounding Box Construction.** The bounding box vertices are constructed with eight vertices from the minimum depth and maximum depth. For qualitative analysis, the edges of the 3D bounding box are projected onto the 2D image plane to visualize the spatial extent of the detected object.

Implementation Details. The projection is applied in a bidirectional manner, mainly because of the lack of data to overview the correctness of the work. Instead of only relying on 3D clustering methods, the algorithm uses the 2D detection box geometry and its depth statistics to its advantage. The height and width of the 3D bounding boxes are supported through estimation from the 2D bounding boxes. This way, the approach creates a middle point between the accurate yet sparse 3D measurements and dense but depth-ambiguous 2D detections.

4.3.3 Simplifications and Assumptions

Several simplifications were adopted to enhance the robustness of the 3D bounding box estimations.

4. Methodology

Point Filtering. Points beyond the distance threshold (3 meters) from the camera, as well as the points that are behind the camera are eliminated to avoid invalid projections.

Height Estimation. The height of the bounding boxes is estimated from the projection of the 2D detection box at the determined depth. This is made as the LiDAR points get more sparse in the higher areas, where the top point of the pole becomes ambiguous. The poles are also assumed to be starting from the ground level for this assumption to work.

Circular Poles. As discussed in the earlier section, the poles are assumed to have circular cross-sections, meaning that the depth and the width of the pole are equal. This was developed to solve the inconsistencies of widths of the poles on some frames.

These simplifications and assumptions allow a more consistent and robust 3D box generation even with sparse point clouds and inherent problems of detecting thin objects like poles.

Chapter 5

Results and Evaluation

This chapter will present the results and evaluate the performance of the proposed detection system. The evaluation begins with the analysis of the 2D detection accuracy of the fine-tuned YOLO model, continues with the 3D localization section and concludes with comparisons and discussions.

5.1 Training Setup and Configuration

The training dataset was composed of annotated pole instances derived from A2D2 semantic segmentation masks and was later converted into YOLO-format bounding box annotations. The training process was configured to run for 60 epochs using a batch size of 16. The input images were resized to 640x640 pixels, which is a common choice that balances GPU memory usage and detection accuracy. The training was done on NVIDIA TITAN Xp, and the GPU provided sufficient performance for training deep neural networks.

5.2 2D Detection Performance

The fine-tuned YOLO model was evaluated on an urban pole detection dataset using a confidence threshold of 0.25. Table 5.1 displays the key performance metrics.

The model exhibits a moderate performance. A precision of 63.3% indicates that most predicted bounding boxes correspond to true poles, reflecting a relatively low false positive rate. The recall at 51.44% signifies that slightly more than half of the

5. Results and Evaluation

Metric	Fine-tuned Model
Precision	0.6333
Recall	0.5138
F1 Score	0.5673
Intersection over Union (IoU)	0.7789
mAP@50	0.4509
Inference Time (s/img)	0.0768

Table 5.1: 2D detection results for the fine-tuned YOLO model (confidence ≥ 0.25).



Figure 5.1: Original Image and Detection

actual poles are detected, suggesting room for improvement in sensitivity, particularly for thinner or occluded poles. The F1 Score of 56.7% suggests a reasonably balanced, although not optimal, detection capability. The average IoU of 0.77% shows that when detections occur, the bounding box detections overlap well with the ground truth, reflecting accurate localization. The mAP@50 score of 45.1% implies that the detections have generally lower confidence levels. Lastly, the inference speed is 0.076 s/image, approximately 14 FPS (NVIDIA TITAN XP), suggesting the model is capable of near real-time performance.

5.2.1 SAHI Evaluation

SAHI[39] was applied to the fine-tuned model and evaluated using the same test dataset. The comparison between the results of the fine-tuned model only and when SAHI is applied can be found in Table 5.2.

The SAHI-enhanced model significantly improves detection performance over only a fine-tuned model. While there is notable improvement in many key metrics, the most substantial improvement is in recall, which shows SAHI’s effectiveness in detecting most targets. The slight decrease in IoU suggests less accurate localization

5. Results and Evaluation

Metric	Model	Model + SAHI	Relative Change
Precision	0.6333	0.7673	+21.17%
Recall	0.5138	0.8742	+70.12%
F1 Score	0.5673	0.8173	+44.11%
IoU	0.7789	0.7411	-4.85%
mAP@50	0.4509	0.4826	+7.03%
Inference Time (s)	0.0768	1.6146	+2001.56%

Table 5.2: Performance comparison of the fine-tuned model with and without SAHI.



Figure 5.2: Detection with SAHI Enhancement

of the detections, likely because of the overlapping slice aggregation. However, the dramatic increase in inference time shows that SAHI is more appropriate for offline applications where detection accuracy is prioritized over speed. Figure 5.2 shows the SAHI detection over the same image as 5.1.

5.3 3D Localization Results

The 3D localization component of the system integrates the 2D bounding box detections into the LiDAR point clouds to generate 3D bounding boxes around generated poles. Although the ground truth 3D bounding boxes for poles were not available, a combination of quantitative and qualitative methods was employed to evaluate the localization accuracy.

The projection can be found in Figure 5.3. Subfigure 5.3a displays the pointcloud before any processing, while Subfigure 5.3b shows the resulting cluster constrained by the 2D detection frustum projection.

RANSAC is applied to the cluster constrained by the 2D detection frustum projection to attain the final pole points. The process is visualized in Figure 5.4.

5. Results and Evaluation

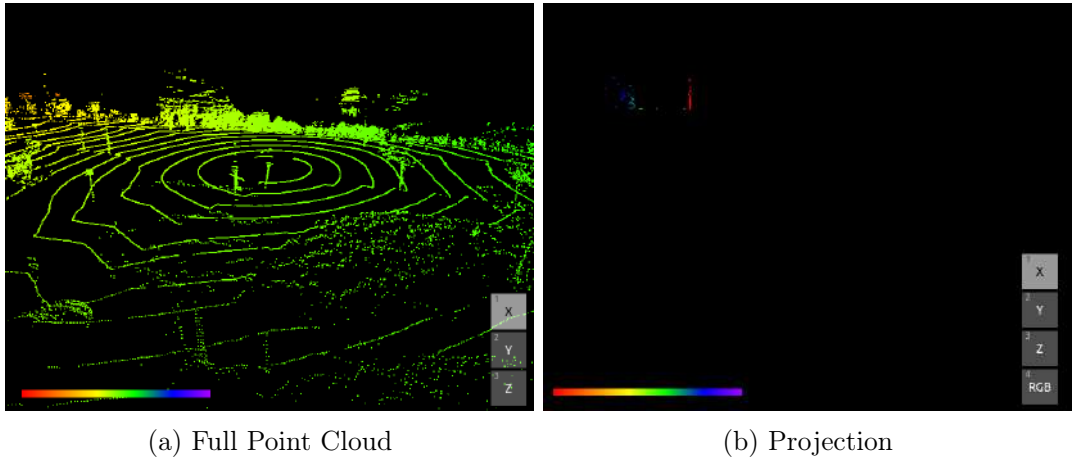


Figure 5.3: Projection of 2D Detection Box

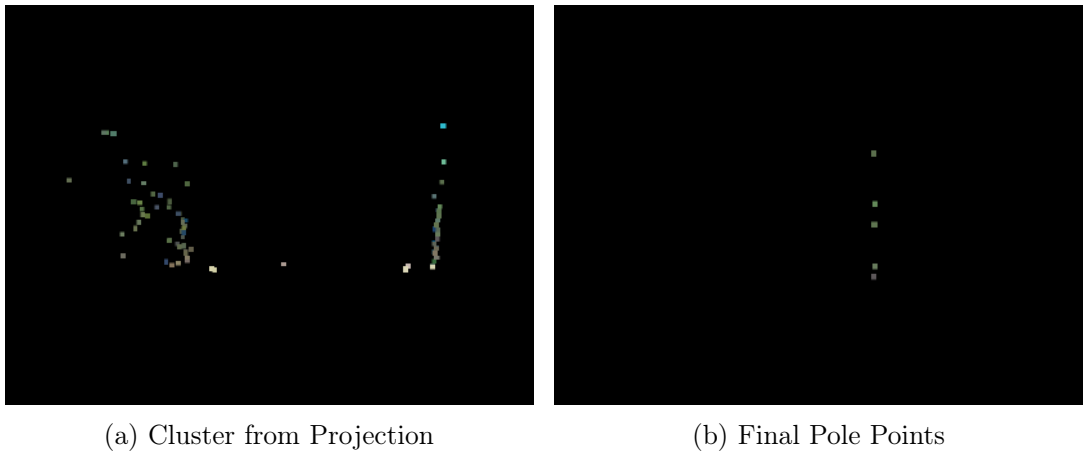


Figure 5.4: Outlier Rejection

Due to the lack of annotated 3D pole data, a simple quantitative analysis was made by manually evaluating 150 frames. The numbers of TP, FP, and FN were counted by manual visual inspection. The results are summarized in Table 5.3.

The overall performance of the detection pipeline is promising, with a satisfactory number of correctly identified poles. A comparatively high number of false negatives reveals a key limitation of the approach: the algorithm occasionally fails to detect poles, particularly those that are exceptionally thin, further away, or partially occluded. This outcome is consistent with expectations, as thin vertical objects span only a few pixels and may not produce a distinct feature that can be reliably detected. Conversely, the rate of false positives remains low, indicating good overall precision. The few instances of false positives are related to tree trunks detected as poles, which exhibit similar features to poles, especially in distance.

5. Results and Evaluation

Table 5.3: 3D Localization Results

Metric	Value
True Positives (TP)	580
False Positives (FP)	11
False Negatives (FN)	169
Precision	0.981
Recall	0.774
F1 Score	0.866

5.3.1 Qualitative Evaluation

A qualitative analysis was carried out by examining the projected 3D bounding boxes in selected image frames. Figure 5.5 presents example frames from the custom data sequence, where final inlier points from the 3D point cloud and their corresponding 3D bounding boxes are projected onto the 2D image plane.

As illustrated in Figure 5.5, as well as in the demonstration video¹, the system performs well overall in detecting and localizing pole-like objects. When a 2D detection box is available, a corresponding 3D bounding box is typically well-aligned with the visible pole in the image, indicating the effectiveness of the 2D-3D pipeline even in the presence of sparse point cloud data.

Some limitations can also be observed. The absence of a 2D detection prevents the generation of a 3D bounding box, leading to missed detections. Additionally, some false positives are also observed in the custom sequence. Although rare, some vertical objects, such as tree trunks, are falsely detected as poles.

Overall, the qualitative analysis demonstrates that the proposed method yields spatially accurate pole detections in urban environments, thereby validating the dual-stage detection pipeline.

5.4 Performance Evaluation

Overall, this project displays promising results even with the lack of models that specialize in 2D or 3D pole detection. The fine-tuned 2D Pole Detector demonstrates good detection accuracy and successfully identifies the majority of the pole instances. When enhanced with the SAHI framework, the 2D detection performance improves significantly, particularly in terms of recall, which increases by over 70%, suggesting

¹Video version can be found in: <https://youtu.be/R0qyMtRLguk>

5. Results and Evaluation

a substantial enhancement in the system's ability to detect smaller and partially occluded pole instances.

The 3D Localization module effectively utilizes 2D detection results to generate accurate 3D bounding boxes around poles, as visually confirmed by the projections shown in Figure 5.5. This indicates that even without explicit 3D annotations, meaningful spatial localization of poles can be achieved through geometric modeling and sensor fusion.

5. Results and Evaluation



Figure 5.5: Sequence displaying the 3D bounding box construction in eight steps.

Chapter 6

Conclusion and Future Work

This chapter summarizes the key contributions of the project, discusses the limitations encountered during the study and presents future improvement suggestions in the field of urban object detection for autonomous navigation systems.

6.1 Summary of Contributions

As outlined in the introduction, this thesis makes three primary contributions.

2D Object Detector for Pole-Like Structures. The project presents a specialized detector by fine-tuning the YOLOv11 architecture. The fine-tuned model is capable of detecting pole objects, which is a class that is typically overlooked in standard object detection datasets. The model achieves a precision of 63.33% and recall of 51.38%, with significant performance improvement when enhanced with SAHI. This showcases how image-based deep learning models, when tailored and augmented, can achieve accurate object detection even for challenging classes.

3D Localization without Annotations. 3D Localization pipeline proposes a sensor fusion approach that projects 2D detection outputs onto LiDAR point clouds to estimate the spatial positioning of pole objects. This pipeline eliminates the need of 3D annotated data, which is a significant challenge for projects involving overlooked classes. The system incorporates point cloud projection, association, outlier rejection, and 3D bounding box construction, connecting image-based detection and spatial understanding. Quantitative analysis reveals a strong F1 score (86.6%) confirming that accurate 3D localization can be achieved with a relatively lightweight

method. This approach also displays a good performance in cluttered and dynamic scenes.

Lightweight Alternative to 3D Object Inference. The project’s methodology demonstrates how combining the strengths of both 2D detections and 3D geometric understanding creates a lightweight alternative to 3D object detection.

6.2 Limitations

Despite the promising results of this project, several limitations have been identified that should be addressed.

Dataset and Model Limitations. The most significant limitation was with the dataset and existing models. The lack of 3D annotated datasets and object detectors for poles has introduced significant challenges to evaluating the proposed model’s performance. Additionally, the preprocessing pipeline introduced potential biases in the bounding box generation process due to the filtering of small and odd-ratio boxes.

SAHI Trade-off. While SAHI-enhancement introduced significant performance improvement, the increase in inference time represents a significant limitation for deployment in real-time projects.

Geometry Assumptions. The assumptions introduced about pole geometry and position may not hold true for some examples, such as misplaced and disfigured poles.

Training Constraints. Limited GPU resources restricted the model architecture exploration and hyperparameter optimization, potentially leaving out some performance improvements.

Absence of Comparative Baseline. The lack of established pole detection model introduced challenges in the evaluation of the model performance.

6.3 Future Work

Based on the contributions and limitations of this study, several directions can be taken to improve and extend this project.

Dataset Expansion. Future work should focus on creating a pole detection dataset with 2D and 3D bounding box annotations, which will improve the performance of the 2D detector training, as well as provide ground truth for performance evaluation of 3D bounding box construction.

Architecture Exploration and Optimization. Investigating alternative detection architectures could potentially improve performance. Furthermore, dedicating more time into hyperparameter optimization can improve the model performance.

Urban object detection field has many opportunities for innovation and improvement. This thesis project has introduced a foundational work for pole detection that can serve as a more advanced pipeline with future improvements, leading the way to more capable and robust autonomous navigation systems in complex urban environments.

Bibliography

- [1] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.
- [2] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV]. URL: <https://arxiv.org/abs/2005.12872>.
- [3] Yian Zhao et al. *DETRs Beat YOLOs on Real-time Object Detection*. 2024. arXiv: 2304.08069 [cs.CV]. URL: <https://arxiv.org/abs/2304.08069>.
- [4] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *European conference on computer vision (ECCV)* (2014).
- [5] Andreas Geiger et al. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2012, pp. 3354–3361.
- [6] Holger Caesar et al. “nuScenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 11621–11631.
- [7] Shimon Ullman. “The interpretation of structure from motion”. In: *Proceedings of the Royal Society of London. Series B. Biological Sciences* 203.1153 (1979), pp. 405–426.
- [8] Arthur Ouaknine. *Review of Deep Learning Algorithms for Object Detection*. Accessed: 2025-04-24. Feb. 2018. URL: <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>.
- [9] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031.

BIBLIOGRAPHY

- [10] Russ Tedrake. *Robotic Manipulation. Perception, Planning, and Control*. 2024. URL: <http://manipulation.mit.edu>.
- [11] Zhengyou Zhang. “Camera Parameters (Intrinsic, Extrinsic)”. In: *Computer Vision: A Reference Guide*. Ed. by Katsushi Ikeuchi. Cham: Springer International Publishing, 2021, pp. 135–140. ISBN: 978-3-030-63416-2. DOI: 10.1007/978-3-030-63416-2_152. URL: https://doi.org/10.1007/978-3-030-63416-2_152.
- [12] Charles R. Qi et al. “Frustum PointNets for 3D Object Detection From RGB-D Data”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.
- [13] Josh Wilson et al. “Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting”. In: *Proceedings of the Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*. 2023.
- [14] Carsten Geyer et al. *A2D2: Audi Autonomous Driving Dataset*. <https://www.a2d2.audi/>. Accessed: YYYY-MM-DD. 2020.
- [15] Oliver Zendel et al. “ZOD: A Driving Dataset for Heterogeneous Multimodal Learning”. In: *arXiv preprint arXiv:2306.08104* (2023).
- [16] Autonomous Mobility and Perception Lab (AMPL). *Website of the Autonomous Mobility and Perception Lab, Universidad Carlos III de Madrid*. <https://ampl.uc3m.es/>. Accessed: 2025-05-23. 2023.
- [17] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. “A Survey on Performance Metrics for Object-Detection Algorithms”. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.
- [18] Hamid Rezatofighi et al. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.
- [19] P. Viola and M. Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

BIBLIOGRAPHY

- [20] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV]. URL: <https://arxiv.org/abs/1311.2524>.
- [21] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: <https://arxiv.org/abs/1506.02640>.
- [22] Jasper Uijlings et al. “Selective Search for Object Recognition”. In: *International Journal of Computer Vision* 104 (Sept. 2013), pp. 154–171. DOI: 10.1007/s11263-013-0620-5.
- [23] Ross Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.
- [24] Muhammad Hussain. “YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection”. In: *Machines* 11.7 (2023), p. 677.
- [25] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *CoRR* abs/1804.02767 (2018). arXiv: 1804.02767. URL: <http://arxiv.org/abs/1804.02767>.
- [26] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: <https://arxiv.org/abs/2004.10934>.
- [27] Glenn Jocher. *YOLOv5 Documentation by Ultralytics*. <https://github.com/ultralytics/yolov5>. Accessed: 2025-04-26. 2020.
- [28] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLOv8 Documentation by Ultralytics*. Accessed: 2025-04-26. 2023. URL: <https://docs.ultralytics.com/models/yolov8/>.
- [29] Chien-Yao Wang and Hong-Yuan Mark Liao. *YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems*. 2024. arXiv: 2408.09332 [cs.CV]. URL: <https://arxiv.org/abs/2408.09332>.
- [30] Glenn Jocher and Jing Qiu. *Ultralytics YOLO11*. Version 11.0.0. 2024. URL: <https://github.com/ultralytics/ultralytics>.

BIBLIOGRAPHY

- [31] Rahima Khanam and Muhammad Hussain. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024. arXiv: 2410.17725 [cs.CV]. URL: <https://arxiv.org/abs/2410.17725>.
- [32] Ankan Ghosh. *YOLO11: Redefining Real-Time Object Detection*. Accessed: 2025-04-26. 2024. URL: <https://learnopencv.com/yolo11/>.
- [33] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.
- [34] Chethan B., R. Punitha, and Mohana . “YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications”. In: Aug. 2020, pp. 1316–1321. DOI: 10.1109/ICSSIT48917.2020.9214094.
- [35] Nandini Reddy. *RT-DETR: The next evolution in real-time object detection*. Accessed: 2025-02-12. 2021. URL: <https://medium.com/@nandinilreddy/rt-detr-the-next-evolution-in-real-time-object-detection-aa1c7880f368>.
- [36] Sourabh Vora et al. *PointPainting: Sequential Fusion for 3D Object Detection*. 2020. arXiv: 1911.10150 [cs.CV]. URL: <https://arxiv.org/abs/1911.10150>.
- [37] Alex H Lang et al. “PointPillars: Fast Encoders for Object Detection from Point Clouds”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12697–12705.
- [38] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: <https://doi.org/10.1145/358669.358692>.
- [39] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. “Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection”. In: *2022 IEEE International Conference on Image Processing (ICIP)* (2022), pp. 966–970. DOI: 10.1109/ICIP46576.2022.9897990.

BIBLIOGRAPHY

- [40] Dawei Du et al. “VisDrone-DET2019: The Vision Meets Drone Object Detection in Image Challenge Results”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2019, pp. 0–0.
- [41] Daniel Lam et al. “xView: Objects in Context in Overhead Imagery”. In: *arXiv preprint arXiv:1802.07856* (2018).
- [42] Sabina Pokhrel. *Exploring Data Labeling and the 6 Different Types of Image Annotation*. Accessed: 2025-04-18. Towards AI. 2020. URL: <https://towardsai.net/p/data-science/exploring-data-labeling-and-the-6-different-types-of-image-annotation>.

Appendix A

Annotation Formats

```

1 {
2   "id": int,
3   "image_id": int,
4   "category_id": int,
5   "segmentation": RLE or [polygon],
6   "area": float,
7   "bbox": [x, y, width, height],
8   "iscrowd": 0 or 1,
9 }
10
11 categories[{
12   "id": int,
13   "name": str,
14   "supercategory": str,
15 }]

```

COCO JSON format. Source: [42]

```

1 <annotation>
2   <folder>Train</folder>
3   <filename>01.png</filename>
4   <path>/path/Train/01.png</path>
5   <source>
6     <database>Unknown</database>
7   </source>
8   <size>
9     <width>224</width>
10    <height>224</height>
11    <depth>3</depth>
12  </size>
13  <segmented>0</segmented>
14  <object>
15    <name>36</name>
16    <pose>Frontal</pose>
17    <truncated>0</truncated>
18    <difficult>0</difficult>
19    <occluded>0</occluded>
20    <bndbox>
21      <xmin>90</xmin>
22      <xmax>190</xmax>
23      <ymin>54</ymin>
24      <ymax>70</ymax>
25    </bndbox>
26  </object>
27 </annotation>

```

Pascal VOC XML format. Source: [42]

A. Annotation Formats

```
1 <object-class> <x> <y> <width> <height>
```

YOLO bounding box format. Source: [42]

```
1 0 45 55 29 67
2 1 99 83 28 44
```

YOLO annotation example. Source: [42]

13

**Enhancing Deep Reinforcement Learning with Curriculum Learning
for Robotic Tasks**
Syma Afsha



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTER IN
INTELLIGENT FIELD ROBOTIC SYSTEMS (IFROS)

Enhancing Deep Reinforcement Learning with Curriculum Learning for Robotic Tasks

Author:

Syma Afsha

B.Sc.in Electrical, Electronic and Communication Engineering

Internal supervisor:

Dr. Zoltán Istenes

Eötvös Loránd University

External supervisor:

Dr. Dániel Horváth

HUN-REN SZTAKI

Budapest, 2025

Abstract

Reinforcement Learning (RL) algorithms often exhibit suboptimal performance in sparse-reward robotic environments due to insufficient exploration, low sample efficiency, and convergence instability. These challenges are particularly critical in robotic systems, where environment interactions are expensive and policy robustness is paramount. Curriculum Learning (CL) has emerged as a principled framework to mitigate these limitations by structuring the learning process. However, most existing CL approaches rely on manually designed curricula or rigid scheduling heuristics, which often fail to generalize across diverse tasks or adapt to learning dynamics.

In this research, I propose novel CL strategies to improve Deep Reinforcement Learning (DRL) in sparse-reward continuous control environments—domains characterized by significant exploration difficulties, high data acquisition costs, and unstable learning dynamics. The proposed Incremental Graded Curriculum (IGC) dynamically adjusts task difficulty based on agent performance, accelerating early learning. In parallel, the Dual Buffer Mechanism (DB) enhances replay efficiency by explicitly separating successful and failed episodes, enabling agents to learn from both experiences. These two components are unified in the Dual-Mechanism Incremental CL (DICL) framework, which integrates progressive task modulation with structured experience replay.

Comprehensive empirical evaluations on three challenging robotic benchmarks—*FetchPickAndPlace*, *FetchPush*, and *FetchSlide*—show that the proposed methods consistently perform better than existing state-of-the-art approaches in terms of convergence speed, sample efficiency, and task success rate. These improvements are difficult to achieve in sparse-reward settings, which highlights the effectiveness and broad applicability of the proposed framework.

Although the results presented in this Master’s Thesis are my own, plural pronouns are used throughout for stylistic consistency.

Acknowledgements

With heartfelt gratitude to the Almighty for providing wisdom and guidance throughout my research, I would like to convey my sincere appreciation for the knowledge shared with me during this journey.

I am deeply grateful to my internal supervisor, Dr. Zoltán Istenes, for his unconditional support and dedication throughout my research. His patient guidance, clear advice, and ongoing commitment have made a huge difference, especially through his constructive feedback. I am also immensely grateful to my external supervisor, Dr. Dániel Horváth, for his inspiring mentorship and dedication. His clear direction, thoughtful feedback, and help in introducing me to new tools have enriched my work and been essential to the success of this research.

I would like to express my gratitude to Intelligent Field Robotic Systems (IFRoS) master program at Universitat de Girona (UdG) and Eötvös Loránd University (ELTE) for offering a supportive academic environment, excellent resources, and collaborative opportunities that have greatly contributed to my personal and professional growth. I am sincerely appreciative of HUN-REN SZTAKI for providing me with the opportunity to pursue this master thesis.

I am really thankful to my family for their unwavering love, encouragement, and support. Their belief in me has been a constant source of strength and inspiration throughout this academic journey.

Finally, I appreciate the assistance and cooperation of my colleagues whose support during challenging times has been invaluable.

This thesis is a tribute to the power of collective effort, divine guidance, and the relentless pursuit of knowledge. Thank you for being an integral part of this transformative journey.

Contents

Abstract	i
Acknowledgements	ii
List of Figures	iv
List of Tables	vi
List of Algorithms	vii
List of Abbreviations	viii
List of Notations	x
1 Introduction	1
1.1 Context	1
1.2 Problem Statement	2
1.3 Research Questions	3
1.4 Contributions	4
1.5 Outline	5
2 Background Theory	6
2.1 Markov Decision Processes	6
2.2 Policy	7
2.3 Value Functions	8
2.4 Single-Goal vs Multi-Goal Tasks	8
2.5 Reward Function	9
2.6 Categorization of RL Algorithms	10
2.7 Tabular Reinforcement Learning	10
2.8 Exploration–Exploitation Trade-off	10

CONTENTS

2.9	Curriculum Learning (CL)	11
2.10	Historical Developments of RL in Robotics	12
3	Related Works	13
3.1	State-of-the-Art RL agents	13
3.2	Curriculum Learning Approach in RL	14
3.2.1	Data Collection	15
3.2.2	Data Exploitation	15
3.3	Research Gap	19
4	Methodology	21
4.1	Incremental Graded Curriculum (IGC)	21
4.2	Dual Buffer Mechanism (DB)	25
4.3	Dual-Mechanism Incremental CL Framework	30
5	Result Analysis	33
5.1	Experimental Setup	33
5.2	Failure States	33
5.3	Training and Evaluation Protocol	34
5.4	Evaluation Metrics	35
5.5	Comparative Performance Evaluation of CL Strategies on Sparse- Reward Robotic Tasks	39
5.5.1	FetchPickAndPlace	39
5.5.2	FetchPush	43
5.5.3	FetchSlide	44
5.6	Performance Overview	46
5.7	Qualitative Evaluation	46
6	Conclusion	47
6.1	Summary of Thesis Achievements	47
6.2	Future Work	50
A	Reinforcement Learning	51
A.1	Categorization of RL Algorithms	51
A.2	Tabular Reinforcement Learning	53
A.2.1	Dynamic Programming	53

CONTENTS

A.2.2	Monte Carlo Methods	54
A.2.3	Temporal-Difference Learning	54
A.3	Circular Replay Buffer	55
A.4	Historical Developments of RL in Robotics	55
B	Soft Actor–Critic (SAC)	57
C	Prioritized Experience Replay (PER)	59
D	Hindsight Experience Replay (HER)	61
	Bibliography	62

List of Figures

2.1	The agent–environment interaction in an MDP.	7
2.2	Overview of Curriculum Learning as a structured training process, showing how tasks progress from simple to complex subtasks [19]. . .	11
2.3	Curriculum learning interpreted as a continuation method: the optimization landscape (top) and policy (bottom) evolve from an easy surrogate objective to the full complex task [19].	12
3.1	Workflow of the Prioritized Experience Replay (PER) algorithm. . . .	16
3.2	Workflow of the Hindsight Experience Replay (HER) algorithm. . . .	16
3.3	Overview of HiER and HiER+ [49]. For every episode, the initial state is sampled from μ_0 . After every episode, the transitions are stored in $\mathcal{B}_{\text{ser}}$, and in case the condition is fulfilled then in $\mathcal{B}_{\text{hier}}$ as well. For training, the transitions are sampled from both $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ according to the ratio ξ	17
4.1	Sampling region expansion in Incremental Graded Curriculum (IGC). The agent begins in the narrow region r_{narrow} , progresses through the intermediate region r_{inter} , and eventually reaches the full space r_{wide} . Object and goal positions are sampled within these progressively expanding regions. The diagram illustrates two objects (black squares) and two goals (red circles), demonstrating how sampling occurs within the current region.	23
4.2	A schematic of the Dual-Mechanism Incremental Curriculum Learning (DICL) framework, where the baseline (model-free) algorithm is shown in yellow , the standard RL workflow and methods (HER, PER) in gray , the Dual-Buffer mechanism (DB) in blue , and the Incremental Goal Curriculum (IGC) module in green . For a detailed description, see Alg. 2.	30

LIST OF FIGURES

5.1	Gymnasium-Robotics Fetch Benchmark Environment Tasks	34
5.2	Comparison between our proposed methods and state-of-the-art baselines based on success rate with 95% confidence intervals. Point estimates are provided in Tab. 5.2.	38
5.3	Evaluation success rate for Pick-and-Place task under three IGC variants and Progression coefficient n for the Fully Adaptive IGC during training.	40
5.4	Comparison of Fixed and Hybrid Adaptive proportional-sampling strategies on the Sliding task and Cumulative number of episodes stored in the three replay buffers during training in the Pick-and-Place task.	40
5.5	ϑ_{pos} changes over time and Evaluation performance between Exponential and Linear predefined thresholds for ϑ_{pos}	41
5.6	Evaluation success rate of Pick-and-Place task compared to the Baseline methods for various agent configurations.	41
5.7	Cumulative reward of Pick-and-Place task compared to the Baseline methods for various agent configurations.	42
5.8	Episode Length of Pick-and-Place task compared to the Baseline methods for various agent configurations.	42
5.9	Evaluation Success Rate of Pushing task compared to the Baseline methods for various agent configurations.	43
5.10	Cumulative reward of Pushing task compared to the Baseline methods for various agent configurations.	43
5.11	Evaluation episode length of Pushing task compared to the Baseline methods for various agent configurations.	44
5.12	Evaluation success rate of Sliding task compared to the Baseline methods for various agent configurations.	45
5.13	Evaluation cumulative reward of Sliding task compared to the Baseline methods for various agent configurations.	45
5.14	Evaluation episode length of Sliding task compared to the Baseline methods for various agent configurations.	46

List of Tables

3.1	Overview of Data Collection Curriculum Learning Methods in RL . .	18
3.2	Overview of Data Exploitation Curriculum Learning Methods in RL .	20
5.1	The Hyperparameters of Baseline SAC	35
5.2	Our proposed method compared to the state-of-the-art methods based on success rate. The best results per task are highlighted in bold. . .	37

List of Algorithms

1	Dual Buffer Mechanism (DB)	29
2	Dual-Mechanism Incremental Curriculum Learning (DICL)	32

List of Abbreviations

AI	Artificial Intelligence
BaRC	Backward Reachability Curriculum
CHER	Curriculum-guided Hindsight Experience Replay
CL	Curriculum Learning
CURIOUS	Intrinsically Motivated Modular Multi-Goal RL
DDPG	Deep Deterministic Policy Gradient
DICL	Dual-Mechanism Incremental Curriculum Learning
DP	Dynamic Programming
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DUP	Dueling Network Architectures
E2H-ISE	Entropy-to-Hardness Initial State Entropy
ERE	Emphasizing Recent Experiences
GOAL GAN	Goal Generative Adversarial Network
HER	Hindsight Experience Replay
HiER	Highlight Experience Replay
IGC	Incremental Graded Curriculum
LSTM	Long Short-Term Memory

List of Abbreviations

MDP	Markov Decision Process
MC	Monte Carlo
PER	Prioritized Experience Replay
PPO	Proximal Policy Optimization
PSB	Proportional Sample Batch scheduling
RL	Reinforcement Learning
SAC	Soft Actor–Critic
SARSA	State–Action–Reward–State–Action
SIL	Self-Imitation Learning
Skew-Fit	State-Covering Self-Supervised RL
TD	Temporal Difference
TD3	Twin Delayed DDPG
UCB	Upper-Confidence Bound

List of Notations

$\mathcal{S}$	State space of the MDP
$\mathcal{A}$	Action space of the MDP
$p(s' \mid s, a)$	Transition probability distribution
$r(s, a)$	Reward function
$\pi(a \mid s)$	Stochastic policy
$\mu(s)$	Deterministic policy
γ	Discount factor
$V^\pi(s)$	State-value function
$Q^\pi(s, a)$	Action-value function
τ	Trajectory: $(s_0, a_0, r_0, s_1, \dots, s_T)$
$\mathcal{D}$	Training Batch
$\mathcal{G}$	Set of possible goals
$\mathcal{H}(\pi)$	Entropy of policy π
α	Entropy coefficient (SAC)
δ	Temporal-Difference (TD) error
R	Return from an episode: $R = \sum_t r_t$
ϑ_{pos}	Positive threshold for reward
ϑ_{neg}	Negative threshold for reward

 List of Notations

$\vartheta_{\text{pos}}^{\min}, \vartheta_{\text{pos}}^{\max}$	Min/max bounds for ϑ_{pos}
ρ	Task sampling range factor (e.g., $\rho_{\text{obj}}, \rho_{\text{goal}}$)
$\rho_{\min}, \rho_{\max}$	Minimum and maximum range factor
n	Normalized curriculum progress coefficient: $n = \frac{\rho - \rho_{\min}}{\rho_{\max} - \rho_{\min}}$
Δ	Step size in self-paced curriculum
$\Upsilon(\psi)$	Logarithmic curriculum progression function
ψ	Success rate of the agent
$\psi_{\min}, \psi_{\max}$	Lower and upper bounds for curriculum adaptation
$\zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}}$	Sampling ratios from primary, positive, and negative buffers
$\zeta_{\min}, \zeta_{\max}$	Sampling bounds for adaptive strategies
ζ_{fix}	Fixed sampling ratio
t	Current training timestep
T	Total number of training steps
τ	Normalized training progress: $\tau = \frac{t}{T}$
$d_{\min}$	Minimum distance between object/goal positions to prevent overlap
o_i, g_j	Object and goal positions
$\delta_i^{(\text{obj})}, \delta_j^{(\text{goal})}$	Offsets sampled from uniform distributions
$u_{\text{obj}}, u_{\text{goal}}$	Centers of object and goal sampling regions
$\mathcal{B}_{\text{ser}}$	Primary standard experience replay buffer
$\mathcal{B}_{\text{pos}}$	Positive replay buffer
$\mathcal{B}_{\text{neg}}$	Negative replay buffer

List of Notations

$\mathcal{B}_{\text{hier}}$	Highlight experience replay buffer
$\mathcal{R}_{\text{penalty}}(t)$	Penalty reward if object dropped
$z_{\text{obj}}(t)$	Object height at time t
z_{thres}	Height threshold for applying penalty
s_{fail}	Failure state

Chapter 1

Introduction

This chapter presents the motivation and background for advancing Deep Reinforcement Learning (DRL) in robotic learning contexts, focusing on addressing core limitations that constrain the effectiveness and scalability of learning. It discusses the role of Curriculum Learning (CL) as a structured framework aimed at enhancing training efficiency, stability, and reliable policy behavior. The chapter formulates the key research questions addressed in this study, presents the primary contributions of the proposed curriculum-based learning strategies, and concludes by outlining the organization of the thesis.

1.1 Context

In today's rapidly evolving landscape of data-driven robotics, static models and pre-programmed algorithms are insufficient to handle the complexity and variability of real-world environments. DRL has emerged as a powerful framework for enabling autonomous agents to learn sophisticated behaviors directly from raw sensory inputs. By integrating deep neural networks with reinforcement learning (RL), DRL has demonstrated remarkable performance across various domains, including chess [1], Go [2], Atari [3], Death match scenarios [4], Dota 2 [5] and real-time strategy battles in StarCraft II [6].

However, despite these advances, the practical application of DRL in robotics remains limited by critical challenges, particularly sample inefficiency and unstable training dynamics. These limitations are amplified in robotic settings, where each interaction with the environment can be costly, time-consuming, or potentially haz-

1. Introduction

ardous. Conventional DRL approaches often require an impractically large number of interactions to learn effective policies [7]. This issue is further compounded by the continuous nature of state and action spaces in robotics and the frequent presence of sparse-reward signals, which together hinder training stability and slow convergence. Consequently, deploying DRL in real-world robotic systems often results in prolonged training durations and suboptimal performance.

Drawing inspiration from human learning processes, where individuals typically acquire knowledge by progressing from simpler concepts to more complex ones, CL [8, 9, 10] has gained prominence as a principled approach to address key limitations of DRL, particularly in robotic applications. Traditionally formulated as an easy-to-hard learning paradigm, CL organizes the training sequence such that agents initially engage with less challenging tasks or environments before gradually advancing to more difficult ones. This progressive structure enhances policy acquisition efficiency, facilitates exploration in sparse-reward settings, and promotes greater training stability.

The conceptual progression from static learning models to DRL, and further to CL-enhanced DRL, reflects a shift toward more adaptive, sample-efficient, and scalable learning frameworks. Owing to its structured alignment with human learning processes, CL presents a promising paradigm for addressing the fundamental limitations of DRL in practical robotic deployments.

1.2 Problem Statement

DRL has shown great potential in robotics, especially for solving complex tasks that involve continuous control. However, applying DRL to robotic systems presents several major challenges. One of the most critical issues is inefficient exploration. DRL algorithms typically require a large number of interactions with the environment to learn effective behaviors. In robotics, each interaction is often time-consuming, costly, and potentially damaging to hardware, making such extensive data requirements impractical. As a result, the agent often fails to discover optimal solutions.

Another significant challenge is the instability of training. The learning process in DRL can be unpredictable and highly sensitive to small changes in parameters, which makes it difficult to train robust models. This problem becomes even more severe

1. Introduction

in continuous control settings, where actions are not discrete and small variations in output can lead to large differences in behavior. In addition, many robotic tasks offer very sparse rewards. This means the agent often receives little feedback during training, making it hard to understand whether its actions are leading to success or failure. As a result, the learning process can become slow and inefficient, with the agent struggling to discover effective strategies on its own.

CL offers a promising approach to address these challenges by structuring the learning process as a progression through tasks of increasing difficulty. Despite its potential, the systematic integration of CL into DRL frameworks remains under-explored. In particular, there is a lack of rigorous comparisons between fixed and adaptive curricula, especially when used in conjunction with structured replay mechanisms. Therefore, there is a clear need to design and evaluate both fixed-schedule and adaptive curriculum strategies, and to investigate how their integration with structured replay can improve sample efficiency, training stability, and overall policy performance in sparse-reward, continuous control environments.

1.3 Research Questions

The effectiveness of CL in DRL critically depends on the systematic design, scheduling, and integration of task progressions within the learning process. Additionally, understanding the interactions between CL strategies, experience replay mechanisms, and exploration behaviors is essential for optimizing learning efficiency and stability. Evaluating the comparative strengths of fixed versus adaptive curricula and assessing their ability to generalize across a variety of tasks are also key to developing robust and scalable DRL systems. To investigate these core aspects, this study formulates the following research questions:

1. How can CL strategies be systematically designed and incorporated into model-free DRL algorithms to maximize sample efficiency, convergence speed, and training stability in continuous-control domains?
2. What are the comparative advantages and trade-offs between fixed-schedule and adaptive curriculum strategies in terms of convergence speed, asymptotic performance, and robustness across standard benchmark tasks?

3. How does the combined use of CL and structured experience replay mechanisms influence the exploration–exploitation balance in continuous-control environments?
4. To what extent do the proposed CL approaches generalize across diverse robotic benchmarks with respect to performance transferability and robustness?

1.4 Contributions

This thesis contributes to the advancement of DRL in robotics by introducing a novel integration of CL and experience replay mechanisms. The key contributions are as follows:

- **Incremental Graded Curriculum (IGC):** A data-collection strategy that progressively expands the object and goal sampling range based on the agent’s performance, guiding the agent through tasks of increasing difficulty to ensure a balanced learning progression.
- **Dual-Buffer Mechanism (DB):** A curriculum-aware experience replay approach that partitions transitions into separate “*positive*” and “*negative*” buffers. This separation allows selective sampling of successes and failures, promoting stable and effective policy updates.
- **Dual-Mechanism Incremental CL (DICL):** A unified framework that integrates IGC and DB into a single pipeline, with optional integration of Hindsight Experience Replay (HER) [11] and Prioritized Experience Replay (PER) [12], to align structured exploration with targeted replay for enhanced efficiency and stability
- **Fixed vs. Adaptive Curriculum Evaluation:** A systematic comparison between fixed-schedule and performance-based adaptive curricula on standard robotic benchmarks, quantifying their respective impacts on convergence, efficiency, and asymptotic task performance.

The experimental results on Gymnasium Robotics Fetch benchmark tasks demonstrate that IGC, DB, and DICL consistently outperform prior baselines in terms of convergence rate, task success metrics, and sample efficiency.

1.5 Outline

This thesis is structured into six chapters, each contributing progressively to the development and validation of the proposed framework:

- **Chapter 1 – Introduction:** Establishes the motivation for the study, outlines key challenges in deploying DRL in robotics, formulates the research questions and objectives, and summarizes the core contributions.
- **Chapter 2 – Background Theory:** Provides foundational knowledge on DRL, experience replay, and CL, setting the theoretical basis for the proposed methodology.
- **Chapter 3 – Related Works:** Critically reviews existing works on curriculum learning and DRL in robotics, highlighting limitations and identifying research gaps that this thesis seeks to address.
- **Chapter 4 – Methodology:** Describes the proposed Dual-Mechanism Incremental Curriculum Learning (DICL) framework in detail, including the design of the curriculum strategies, and replay mechanisms.
- **Chapter 5 – Result Analysis:** Presents the experimental results, comparing the performance of the proposed curriculum configurations against baseline methods, and analyzing their effects on learning efficiency and robustness.
- **Chapter 6 – Conclusion:** Summarizes the key findings, discusses the implications of the results, and proposes directions for future research.

Chapter 2

Background Theory

This chapter presents a structured overview of DRL with a focus on its application to robotics. It introduces by formalizing the standard RL framework and identifying the key challenges associated with sample inefficiency and sparse rewards. CL is then described as a principled strategy to address these limitations by guiding exploration through structured task progression. The chapter concludes by reviewing the development of RL in robotic systems, highlighting the transition from conventional adaptive controllers to contemporary approaches based on policy gradients and demonstration learning.

2.1 Markov Decision Processes

Following [13], a RL problem is formalized as a Markov Decision Process (MDP), defined by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r), \quad (2.1)$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $p(s' | s, a)$ the transition probability density of moving to s' from s under action a , and $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the immediate reward function. At each discrete time t , the agent in state s_t selects an action a_t , then transitions according to

$$s_{t+1} \sim p(\cdot | s_t, a_t), \quad r_{t+1} = r(s_t, a_t). \quad (2.2)$$

The *Markov property* asserts that the probability of transitioning to the next state and receiving a given reward depends only on the current state and action, and is conditionally independent of all preceding history. Fig. 2.1 illustrates this

2. Background Theory

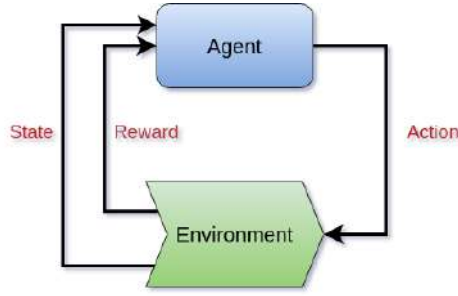


Figure 2.1: The agent–environment interaction in an MDP.

agent–environment loop, which serves as the foundational framework for both model-free and model-based DRL algorithms.

2.2 Policy

A *stochastic policy* π defines a probability distribution over actions given a state:

$$\pi(a \mid s) = \Pr(A_t = a \mid S_t = s), \quad \sum_{a \in \mathcal{A}} \pi(a \mid s) = 1, \quad (2.3)$$

where $\mathcal{A}$ denotes the discrete action space.

In contrast, a *deterministic policy* μ maps each state to a specific action with certainty:

$$\mu(s) = a, \quad (2.4)$$

implying that $\pi(a \mid s) = 1$ and $\pi(a' \mid s) = 0$ for all $a' \neq a$.

In discrete domains, a deterministic policy can often be obtained by selecting the most probable action under a stochastic policy:

$$\mu(s) = \arg \max_a \pi(a \mid s). \quad (2.5)$$

In *continuous action spaces*, where $\mathcal{A} \subset \mathbb{R}^n$, the set of possible actions is uncountably infinite, making explicit enumeration infeasible and rendering the $\arg \max$ operator generally inapplicable. Nevertheless, deterministic policies remain well-defined as functions:

$$\mu : \mathcal{S} \rightarrow \mathcal{A}, \quad a = \mu(s), \quad (2.6)$$

which directly output continuous actions given a state.

The objective in RL is to learn an optimal policy π^* (or μ^*) that maximizes the

2. Background Theory

expected cumulative return:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right], \quad (2.7)$$

where $\gamma \in [0, 1)$ is the discount factor. Policy optimization involves adjusting the policy parameters—either through policy gradient methods or actor-critic algorithms—to improve the expected return $J(\pi)$.

2.3 Value Functions

Value functions quantify an agent's expected future reward under policy π . The state-value function

$$V^{\pi}(s) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right] \quad (2.8)$$

measures the return from state s , while the action-value function

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a \right] \quad (2.9)$$

evaluates the return when taking a in s . They satisfy the Bellman expectation equations:

$$V^{\pi}(s) = \sum_a \pi(a \mid s) \sum_{s'} p(s' \mid s, a) [r(s, a) + \gamma V^{\pi}(s')], \quad (2.10)$$

$$Q^{\pi}(s, a) = \sum_{s'} p(s' \mid s, a) \left[r(s, a) + \gamma \sum_{a'} \pi(a' \mid s') Q^{\pi}(s', a') \right]. \quad (2.11)$$

The optimal functions V^* and Q^* obey the Bellman optimality equations:

$$V^*(s) = \max_a Q^*(s, a), \quad (2.12)$$

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s'} p(s' \mid s, a) \max_{a'} Q^*(s', a'). \quad (2.13)$$

2.4 Single-Goal vs Multi-Goal Tasks

Reinforcement learning tasks differ fundamentally in the number and variability of goals, which in turn shapes policy design, exploration, and sample efficiency.

2. Background Theory

Single-Goal Tasks: In this task, the agent pursues one fixed goal g throughout training and evaluation. This setup often leads to efficient benchmarking but limited generalization beyond the trained goal.

Multi-Goal Tasks: In multi-goal reinforcement learning, the goal g is sampled from a set $\mathcal{G} \subseteq S$ at the beginning of each episode [14], and the policy $\pi(a \mid s, g)$ must generalize across all goals in $\mathcal{G}$ [15]. Each goal corresponds to a set of desired goal states $S_g \subseteq S$, and the task is considered successful when the agent reaches any state $s_t \in S_g$. While multi-goal formulations introduce higher variance in the reward signal due to task diversity [16], they enable learning a single, universal policy capable of achieving any target in the goal set. This property is particularly advantageous in complex robotic tasks, where flexibility and generalization across diverse objectives are essential.

2.5 Reward Function

The reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ defines the agent’s objective by mapping each state–action pair (s, a) to a scalar reward $r(s, a)$. This immediate feedback guides the agent toward behaviors that maximize the expected cumulative return. However, an ill-specified reward can lead to suboptimal or unintended behaviors, ultimately slowing or preventing learning.

Sparse Rewards. In many robotic and goal-oriented tasks, rewards are *sparse*—the agent receives feedback only upon achieving a goal or encountering failure. A typical sparse-reward function is defined as:

$$r_t = \begin{cases} 0, & \text{if } s_t \in S_g, \\ -1, & \text{otherwise.} \end{cases} \quad (2.14)$$

Such binary signals provide limited learning guidance, often resulting in severe sample inefficiency and unstable policy updates, as positive rewards are rare and delayed.

2.6 Categorization of RL Algorithms

This section provides a structured overview of the major categories of RL algorithms. Specifically, it distinguishes between model-based and model-free approaches, on-policy and off-policy learning, and the core algorithmic classes: value-based, policy-based, and actor–critic methods. These distinctions are crucial for understanding the underlying mechanisms, trade-offs, and applicability of various RL strategies, particularly in continuous-control robotic environments. While the primary focus here is to introduce these categories and highlight representative algorithms, more detailed insights and implementation considerations are provided in Appendix A.1.

2.7 Tabular Reinforcement Learning

Tabular reinforcement learning addresses problems with discrete and tractable state-action spaces, where value functions can be explicitly stored and updated in tabulated form. Classic methods in this setting include Dynamic Programming (DP) [17], Monte Carlo (MC) [13], and Temporal Difference (TD) [18], all of which rely on value function estimation. A brief summary appears in App A; for comprehensive treatment, see Sutton and Barto [13].

2.8 Exploration–Exploitation Trade-off

RL agents must balance *exploitation*, using actions known to yield high rewards, and *exploration*, trying new actions that might lead to even better outcomes. Excessive exploitation can trap the agent in suboptimal behavior, while too much exploration wastes time on unpromising moves. Simple strategies, such as occasionally choosing a random action, sampling actions based on their estimated value, or giving bonuses to seldom-tried actions, help maintain this balance. Tuning these strategies over time ensures the agent quickly learns effective long-term policies.

2.9 Curriculum Learning (CL)

Curriculum Learning (CL) is inspired by educational psychology, where humans acquire knowledge progressively by mastering simpler concepts before advancing to more complex ones. In machine learning, early studies in supervised learning demonstrated that training models on ordered sequences of samples—typically from easy to difficult—can significantly enhance convergence speed and generalization ability [8, 9]. A typical CL framework consists of two main components: a *difficulty estimator* that assigns complexity levels to data or tasks, and a *training scheduler* that determines the order in which they are presented. When applied to RL, the learning process occurs across dimensions such as goal difficulty, reward sparsity, or environmental complexity [19]. Since RL agents collect their own training data through environment interaction, CL in RL can influence either the data collection process or the data exploitation process. In the former, curricula are implemented by manipulating the initial state distributions [20, 21], goal configurations [11, 22], reward functions [23, 24, 25], or environmental parameters such as obstacle density [26]. In the latter, curriculum strategies intervene in the replay buffer by prioritizing or relabeling transitions to emphasize more informative experiences during training [9].

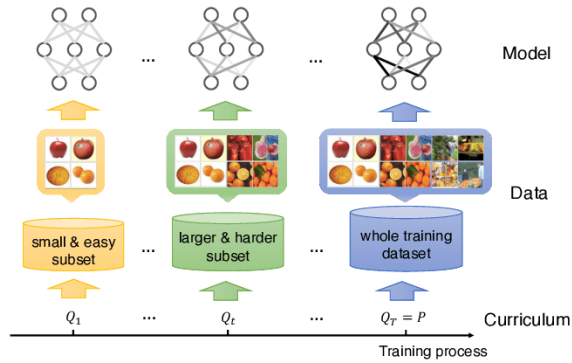


Figure 2.2: Overview of Curriculum Learning as a structured training process, showing how tasks progress from simple to complex subtasks [19].

As shown in Fig. 2.2, agents often begin with simplified tasks, such as reaching nearby goals or operating in obstacle-free environments, and progressively transition to more challenging scenarios. This structure is formalized through continuation methods, which begin with a smoother surrogate objective and gradually transform it into the full target task [27, 19]. This process shapes the optimization landscape in a way that smooths early-stage learning and helps avoid convergence to suboptimal

2. Background Theory

local minima, guiding the policy toward more effective solutions. Fig. 2.3 visualizes this transition from simplified to complex objectives.

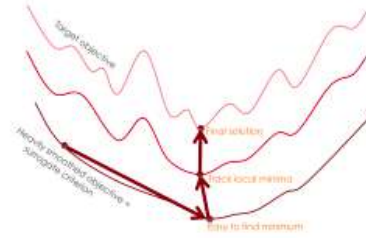


Figure 2.3: Curriculum learning interpreted as a continuation method: the optimization landscape (top) and policy (bottom) evolve from an easy surrogate objective to the full complex task [19].

Curricula can be either fixed or adaptive. Fixed curricula follow a predefined progression, often designed based on domain expertise or heuristics [28], while adaptive curricula dynamically adjust difficulty based on measures such as agent competence, learning progress, or reward feedback [29, 30]. Common implementation techniques include task scheduling, reward shaping for intermediate feedback [24], state-initialization control to guide exploration [31], and gradual parameterization of environment complexity [26].

Although the easy-to-hard paradigm remains the most prevalent CL strategy [8], it is not universally optimal. Alternative approaches such as hard-to-easy, reverse curricula [20], and hard example mining [32] have proven effective in specific settings. As a result, the notion of curriculum has been generalized to any ordered sequence of training criteria—regardless of whether difficulty increases or decreases over time—that facilitates learning and improves performance [9]. In sparse-reward domains such as robotics, CL has consistently demonstrated benefits in accelerating convergence, improving stability, and enabling robust policy learning suitable for real-world deployment [20, 9].

2.10 Historical Developments of RL in Robotics

The historical progression of RL in robotics demonstrates a systematic advancement toward sample-efficient, scalable, and generalizable learning frameworks, ultimately enabling autonomous agents to acquire complex behaviors directly from raw interactions within real-world environments, as outlined in Appendix A.4.

Chapter 3

Related Works

This chapter outlines key advancements in DRL, focusing on methods such as experience replay and CL in robotics. It reviews methods including Prioritized Experience Replay (PER), Hindsight Experience Replay (HER), and Hierarchical Experience Replay (HiER), highlighting their roles in improving sample efficiency and stability. Curriculum strategies showcase how various methods structure learning progression. Finally, the review identifies the research gap in the existing work and highlights areas for further advancement in robotic applications.

3.1 State-of-the-Art RL agents

Selecting an appropriate RL algorithm is crucial for achieving stable and sample-efficient learning in robotics, particularly in environments characterized by sparse rewards and continuous state-action spaces.

Proximal Policy Optimization (PPO) [33] is a widely adopted on-policy method that achieves training stability through a clipped objective function, which constrains policy updates to prevent divergence. Although PPO performs well in many benchmark environments due to its robustness, its on-policy nature significantly reduces sample efficiency. In robotic settings, where data collection is expensive and time-consuming, this inefficiency hinders scalability and learning speed [34, 35]. Deep Deterministic Policy Gradient (DDPG) algorithm [36] is an off-policy actor-critic method designed for continuous control. It combines deterministic policy gradients with experience replay to enable efficient learning. However, empirical studies [37] have shown that DDPG is highly sensitive to hyperparameter settings

3. Related Works

and prone to unstable training. Moreover, its deterministic policy limits exploration, which becomes especially problematic in sparse-reward environments where broad and consistent exploration is essential. Twin Delayed DDPG (TD3) [38] improves upon DDPG by introducing techniques such as clipped double Q-learning, delayed policy updates, and target policy smoothing. While TD3 exhibits improved stability and performance in dense-reward settings, it still employs deterministic policies and consequently suffers from limited exploration. This limitation is particularly detrimental in high-dimensional, sparse-reward robotic tasks, where discovering meaningful trajectories requires robust stochastic exploration [34].

In contrast, Soft Actor-Critic (SAC) [39] integrates stochastic policy optimization with entropy maximization in an off-policy learning framework. Its objective function includes an entropy term that promotes diverse action selection by maximizing both expected return and policy entropy. This design effectively addresses the exploration–exploitation trade-off. SAC also benefits from experience replay, which enhances sample efficiency, and entropy regularization, which prevents premature convergence to suboptimal deterministic policies. The core implementation details of the SAC algorithm are described in Appendix B. Empirical studies have demonstrated that SAC achieves faster convergence, improved policy stability, and better generalization in high-dimensional tasks with sparse-reward signals [40, 41, 21, 42, 43].

Considering the limitations of alternative RL agents and the advantages offered by entropy-regularized stochastic exploration, SAC is adopted in this research as the primary algorithm for training agents in continuous robotic domains, owing to its strong theoretical foundations and consistently robust performance across benchmark evaluations.

3.2 Curriculum Learning Approach in RL

By guiding the agent through a sequence of progressively challenging tasks, CL facilitates the acquisition of complex behaviors through an ordered learning process. Curriculum learning approaches in RL are broadly categorized into two major classes: *data collection curricula* and *data exploitation curricula* [26].

3.2.1 Data Collection

Reverse curriculum generation [20] initializes episodes near the goal and progressively shifts start states farther away as the agent’s performance improves, thereby increasing task difficulty. Ivanovic et al. [44] propose BaRC, which uses approximate backward reachable sets to adaptively sample initial states and control task difficulty in high-dimensional environments. Salimans and Chen [45] seed training with human demonstration states, exposing the agent to proven trajectories to accelerate convergence and improve stability. Portelas et al. [26] dynamically reshape the initial-state distribution using real-time performance feedback to sample only informative, safe states and mitigate catastrophic failures. In teacher–student frameworks [27], a teacher module selects easier start states first and progressively shifts to harder ones as the agent’s proficiency grows.

Racanière et al. [46] use solver feedback to shape a multi-goal setter, adapting the goal distribution toward tasks that are both challenging and achievable. Skew-Fit [21] biases goal sampling toward underexplored states by skewing probabilities inversely with visitation frequency, promoting balanced exploration. Goal GAN [29] trains a generator–discriminator pair to propose goals of intermediate difficulty, neither too easy nor too hard, by learning from the agent’s success signals. Meta-ACL [47] optimizes goal curricula across multiple agents, leveraging past performance data to produce a generalized and transferable curriculum. CURIOUS [48] ranks goals by learning progress and samples those with maximal improvement, keeping the curriculum at the frontier of the agent’s capabilities. Colas et al. [48] track per-goal learning progress and sample goals with the highest improvement rates, deprioritizing those that are too easy or too difficult to maintain a frontier-focused curriculum. Horváth et al. [49] regulate the entropy $H(\mu_0)$ of the joint initial-state–goal distribution, starting nearly deterministic (easy tasks) and gradually increasing entropy to diversify and harden tasks in line with the agent’s capabilities, depicted in 3.3 (green one).

3.2.2 Data Exploitation

Several alternative curriculum-based exploitation methods have also been explored. One such method, PER [33] modifies standard experience replay $\mathcal{B}_{\text{ser}}$ by prioritizing transitions with large high temporal-difference (TD) error, enabling the

3. Related Works

agent to focus on the most informative experiences (see Fig. 3.1 for the workflow). Appendix C provides a detailed explanation of the PER workflow, illustrating how TD error-based sampling improves sample efficiency by focusing updates on high-learning-potential transitions.

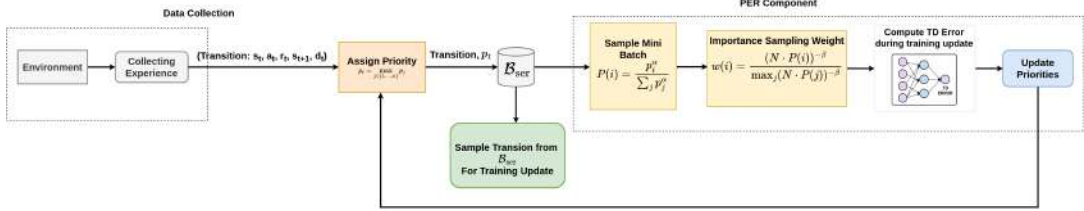


Figure 3.1: Workflow of the Prioritized Experience Replay (PER) algorithm.

Alternative methods for data exploitation curriculum include goal relabeling techniques such as HER [11]. As shown in Fig. 3.2, HER is a replay strategy for sparse-reward reinforcement learning that transforms failed experiences into useful training samples by relabeling goals post-episode. It enables agents to learn from outcomes they actually achieved, thereby improving sample efficiency. A more detailed description of HER is provided in Appendix D.

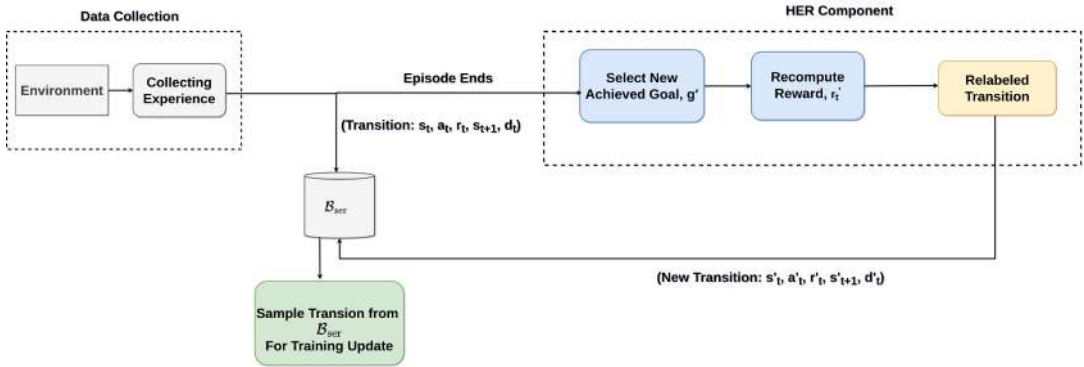


Figure 3.2: Workflow of the Hindsight Experience Replay (HER) algorithm.

Oh et al. [50] introduced Self-Imitation Learning (SIL) as a prioritization-based method within on-policy reinforcement learning frameworks. SIL computes priorities by measuring the advantage of past trajectories using discounted cumulative returns and reinforces those experiences where the advantage is positive. By applying a clipped advantage function, the algorithm selectively amplifies high-return episodes, enabling the agent to repeatedly learn from its own prior successes. Curriculum-guided Hindsight Experience Replay (CHER) [51] introduces a dynamic curriculum

3. Related Works

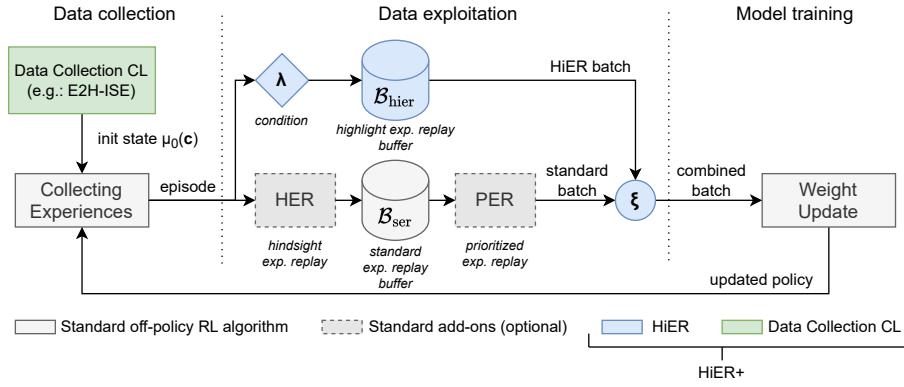


Figure 3.3: Overview of HiER and HiER+ [49]. For every episode, the initial state is sampled from μ_0 . After every episode, the transitions are stored in $\mathcal{B}_{\text{ser}}$, and in case the condition is fulfilled then in $\mathcal{B}_{\text{hier}}$ as well. For training, the transitions are sampled from both $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ according to the ratio ξ .

by adaptively selecting failed experiences based on goal proximity and curiosity-driven diversity. By gradually shifting from exploratory replay to goal-focused sampling, CHER significantly improves learning efficiency.

Reward relabeling techniques [52] further optimize the replay process by assigning bonus rewards to specific transitions within an episode, emphasizing sequences that lead to high returns. Additionally, Self-Imitation Advantage Learning [53] extends SIL to off-policy settings by modifying the Bellman operator, broadening the utility of self-imitation across different RL paradigms. Recently, Emphasizing Recent Experience (ERE) [54] has also been proposed, which prioritizes the most recent interactions while ensuring historical data is retained.

Horváth et al. [49] proposed Highlight Experience Replay (HiER), a data exploitation curriculum learning method that introduces to emphasize *highlighted* transitions. In this approach, the standard replay buffer is complemented by a secondary *highlight* buffer (blue) that selectively stores entire episodes whose cumulative rewards exceed a threshold, illustrated in Fig. 3.3.

3. Related Works

Table 3.1: Overview of Data Collection Curriculum Learning Methods in RL

Method	Key Description
Initial State Distribution	
Reverse Curriculum Generation [20]	Initial state is close to the goal. The distance to the goal is gradually increased, thereby raising task difficulty.
BaRC [44]	Approximate backward reachable sets guide sampling of learnable initial states. This provides a robust way to increase task difficulty in high-dimensional spaces.
Demonstration-Based Initialization [45]	Initial state is sampled from human demonstrations, ensuring early success and faster convergence.
Dynamic Reshaping [26]	The initial state distribution is adapted based on real-time performance metrics, reducing catastrophic failures.
Teacher-Student Curriculum [27]	A teacher module selects easier starting states initially and gradually introduces harder ones as the agent’s proficiency grows.
Goals	
Goal GAN [29]	A generator is trained to output goals of intermediate difficulty, guided by a discriminator for feasibility.
Skew-Fit [21]	Entropy of the goal-conditioned visited states is maximized by skewing sampling toward underexplored areas.
Meta-ACL [47]	Goal sampling is optimized across multiple agents, enabling a more generalized curriculum.
CURIOUS [48]	Goals with higher learning progress are prioritized, ensuring the agent faces appropriately challenging tasks.
Setter-Solver [46]	A setter agent adapts the goal distribution based on the solver’s performance, maintaining steady progress.
Initial State–Goal Distribution	
E2H-ISE [49]	Controls the entropy $H(\mu_0)$ of the joint distribution, starting near-deterministic and gradually expanding to harder tasks.
IGC (Ours)	Updates the task difficulty range for both the initial state and the goal independently, transitioning from narrowly aligned to widely separated pairs.

3.3 Research Gap

Although there are several advancements in CL for DRL shown in Tab. 3.1 and Tab. 3.2, existing methods still leave critical gaps in both *data-collection* and *data-exploitation* curricula. In general, data-collection curricula gradually increase task difficulty by shaping how states and goals are sampled, while data-exploitation curricula determine which past experiences are replayed to the agent. Among the works surveyed, the HiER paper is closest to our approach, since it also seeks to bias replay towards *useful* experiences.

- **Granularity of task-difficulty control:** Early curriculum strategies typically rely on a single global parameter to modulate task difficulty, which limits the ability to perform fine-grained adjustments:
 - *E2H-ISE* adjusts the entropy of the joint initial-state-goal distribution $H(\mu_0)$ using a scalar parameter $c \in [0, 1]$. This produces a smooth *easy-to-hard* progression, but couples the sampling of initial states and goals into a single degree of freedom.
 - *IGC*, by contrast, introduces decoupled sampling ranges for initial states and goals using independent scaling factors ρ_{obj} and ρ_{goal} . This design enables fine-grained, dimension-wise control over task complexity and facilitates more structured exploration.
- **Balance of positive vs. negative experiences in replay:** Many existing replay-based curriculum strategies predominantly emphasize successful episodes, often overlooking the instructive value of failures:
 - *HiER* maintains a secondary buffer $\mathcal{B}_{\text{hier}}$ that only stores episodes whose return R exceeds a threshold λ , thereby neglecting failures.
 - *DB* partitions experiences into three pools—primary $\mathcal{B}_{\text{ser}}$, positive $\mathcal{B}_{\text{pos}}$ ($R \geq \vartheta_{\text{pos}}$), and negative $\mathcal{B}_{\text{neg}}$ ($R \leq \vartheta_{\text{neg}}$)—and samples from each with fixed or adaptive ratios, explicitly reinforcing successes while still learning from failures.

To bridge these gaps, we introduce the *Dual-Mechanism Incremental Curriculum Learning (DICL)* framework, which:

3. Related Works

- *Structures exploration* via IGC’s decoupled expansion of initial-state and goal sampling regions, adapting task complexity to the agent’s current performance level.
- *Shapes exploitation* through DB’s balanced replay from both positive and negative buffers, maintaining diversity and preventing over-emphasis on only high-return experiences.

Compared to the prior E2H-ISE with HiER combination, DICL offers richer task parameterization and more nuanced experience replay, yielding faster convergence, higher final success rates, and more stable learning dynamics on sparse-reward robotic benchmarks.

Table 3.2: Overview of Data Exploitation Curriculum Learning Methods in RL

Method	Key Description
PER [12]	Prioritizes transitions with high temporal-difference (TD) errors to focus updates on informative experiences.
HER [11]	Relabels failed episodes with alternative goals to transform them into successful learning examples.
SIL [50]	Replays past trajectories with positive advantage values to reinforce prior successful behavior.
CHER [51]	Guides hindsight replay using a curriculum based on goal proximity and diversity to improve learning efficiency.
Reward Relabeling [52]	Adds bonus rewards to important transitions to emphasize high-return behaviors during replay.
Self-Imitation Advantage Learning [53]	Extends self-imitation to off-policy settings by modifying the Bellman update to prioritize successful transitions.
ERE [54]	Prioritizes recent transitions to maintain adaptive learning while preserving older experiences.
HiER [49]	Adds a secondary buffer to emphasize high-reward experiences.
DB Mechanism (Ours)	Introduces dual buffers for positive and negative transitions, allowing the agent to learn from both success and failure.

Chapter 4

Methodology

In this chapter, we present our proposed methods to improve learning efficiency in sparse-reward robotic tasks through CL. Section 4.1 introduces the **Incremental Graded Curriculum (IGC)** for structured data collection, and Section 4.2 presents the **Dual Buffer Mechanism (DB)** for curriculum-driven experience replay. Each method includes multiple variants, which are described in detail in their respective sections. Finally, we integrate these components into a unified framework, the **Dual-Mechanism Incremental Curriculum Learning (DICL)** framework, which is presented in Section 4.3. This framework leverages structured exploration and experience replay to enhance learning across varying task complexities.

4.1 Incremental Graded Curriculum (IGC)

We propose *Incremental Graded Curriculum (IGC)*, a novel data collection CL strategy that improves sample efficiency by progressively increasing task difficulty based on agent performance. The training process is initialized with simple tasks, where objects and goals are placed within a narrow sampling region denoted as r_{narrow} . This setup reduces complexity and improves the likelihood of early success, particularly in sparse-reward environments. Even when initial performance is low, the constrained region increases the chance of meaningful feedback compared to sampling over the full workspace. These early successes help the agent acquire essential motor skills and form a foundation for more complex behaviors.

As training progresses, the sampling region expands from narrow (r_{narrow}) to intermediate (r_{inter}) and eventually to wide (r_{wide}), covering the full environment.

4. Methodology

Here, r_{inter} represents a transitional region between r_{narrow} and r_{wide} , rather than a fixed boundary, enabling continuous adaptation as the agent improves.

To regulate this expansion, a logarithmic schedule is used during the intermediate phase. Compared to linear or quadratic progressions, the logarithmic function increases more gradually at first, allowing the agent to spend more time mastering simpler tasks before advancing to more complex ones. This prevents premature failure and supports smoother policy development.

During training, the positions of objects $o_i \in \mathbb{R}^N$ and goals $g_j \in \mathbb{R}^N$ are sampled from uniform distributions centered at $u_{\text{obj}} \in \mathbb{R}^N$ and $u_{\text{goal}} \in \mathbb{R}^N$, respectively:

$$o_i = u_{\text{obj}} + \delta_i^{(\text{obj})}, \quad \delta_i^{(\text{obj})} \sim \mathcal{U}(-\rho_{\text{obj}}, \rho_{\text{obj}}), \quad i = 1, \dots, M \quad (4.1)$$

$$g_j = u_{\text{goal}} + \delta_j^{(\text{goal})}, \quad \delta_j^{(\text{goal})} \sim \mathcal{U}(-\rho_{\text{goal}}, \rho_{\text{goal}}), \quad j = 1, \dots, K \quad (4.2)$$

Here, M and K denote the number of objects and goals, respectively. The parameters $\rho_{\text{obj}} \in \mathbb{R}^N$ and $\rho_{\text{goal}} \in \mathbb{R}^N$ represent the current range factors, specifying the half-widths of the uniform sampling intervals along each dimension of the object and goal spaces. These range factors are dynamically adjusted based on the agent's performance, enabling fine-grained control over task difficulty in each spatial dimension. When both are set to their minimum value ρ_{min} , the agent is in the narrow region r_{narrow} . If they lie between ρ_{min} and ρ_{max} , the agent is in the intermediate region r_{inter} . Once both reach ρ_{max} , the agent enters the wide region r_{wide} , facing the most complex task instances, as illustrated in Fig. 4.1

To prevent overlapping placements and ensure spatial diversity, sampled offsets must satisfy:

$$\|\delta_p - \delta_q\|_2 \geq d_{\text{min}}, \quad \forall p, q \in \{1, \dots, M + K\}, p \neq q \quad (4.3)$$

where $\delta_p, \delta_q \in \mathbb{R}^N$ are the offset vectors, and d_{min} is a predefined minimum distance. This promotes interaction with distinct configurations and supports robust policy learning.

To quantify curriculum progress and modulate task difficulty, we define a normalized curriculum progress coefficient $n \in [0, 1]$ as:

$$n = \frac{\rho - \rho_{\text{min}}}{\rho_{\text{max}} - \rho_{\text{min}}} \quad (4.4)$$

where ρ is the current range factor (e.g., ρ_{obj} or ρ_{goal}).

4. Methodology

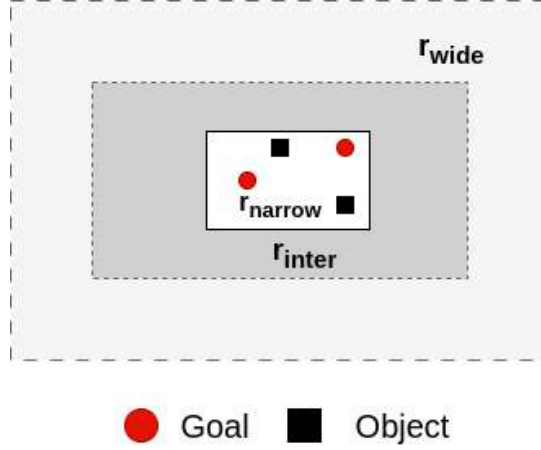


Figure 4.1: Sampling region expansion in Incremental Graded Curriculum (IGC). The agent begins in the narrow region r_{narrow} , progresses through the intermediate region r_{inter} , and eventually reaches the full space r_{wide} . Object and goal positions are sampled within these progressively expanding regions. The diagram illustrates two objects (black squares) and two goals (red circles), demonstrating how sampling occurs within the current region.

Based on n , the environment transitions between curriculum stages:

$$r(n) = \begin{cases} r_{\text{narrow}}, & \text{if } n = 0 \\ r_{\text{inter}}, & \text{if } 0 < n < 1 \\ r_{\text{wide}}, & \text{if } n = 1 \end{cases} \quad (4.5)$$

This structured and interpretable framework for increasing task difficulty is generalizable across environments and task types.

We implement three variants of IGC to control task progression during training. Two of these—*Fully Adaptive* and *Self-Paced*—use a logarithmic progression function, Υ to interpolate between range factor bounds $(\psi_{\min}, \psi_{\max})$ based on success rate, ψ :

$$\Upsilon(\psi) = \frac{\log(1 + \hat{\psi})}{\log(2)}, \quad \text{with} \quad \hat{\psi} = \frac{\psi - \psi_{\text{lower}}}{\psi_{\text{upper}} - \psi_{\text{lower}}}.$$

• Fully Adaptive Curriculum (Logarithmic)

In the fully adaptive strategy, the sampling range factor $\rho(\psi)$ is updated continuously using the agent’s recent success rate $\psi \in [0, 1]$. When the agent performs poorly ($\psi \leq \psi_{\text{lower}}$), the sampling range is kept minimal ($\rho = \rho_{\min}$), creating a narrow-range curriculum. When the agent performs well ($\psi \geq \psi_{\text{upper}}$), the range is expanded to its maximum value ($\rho = \rho_{\max}$), resulting in a wide-range

4. Methodology

curriculum. For intermediate performance ($\psi_{lower} < \psi < \psi_{upper}$), we interpolate smoothly between the two using a logarithmic progression:

$$\rho(\psi) = \begin{cases} \rho_{\min}, & \text{if } \psi \leq \psi_{lower} \\ \rho_{\min} + (\rho_{\max} - \rho_{\min}) \cdot \Upsilon(\psi), & \text{if } \psi_{lower} < \psi < \psi_{upper} \\ \rho_{\max}, & \text{if } \psi \geq \psi_{upper} \end{cases} \quad (4.6)$$

This approach allows the curriculum to adapt in real-time based on the agent’s actual performance.

- **Self-Paced Curriculum (Logarithmic, Non-Decreasing)**

In the *self-paced strategy*, task difficulty evolves based on the agent’s success rate but never decreases. When the agent performs well ($\psi \geq \psi_{upper}$), the current range factor ρ_{prev} is increased by a small step Δ , but is limited to not exceed the maximum range $\rho_{\max}$. If the agent is not performing well, the range is updated using a logarithmic schedule. However, to make sure the difficulty never decreases, the new range is not allowed to go below the highest difficulty reached so far, which is tracked as ρ_{achieved} . This update mechanism is formally defined as:

$$\rho(\psi) = \begin{cases} \min(\rho_{\text{prev}} + \Delta, \rho_{\max}), & \text{if } \psi \geq \psi_{upper} \\ \max(\rho_{\text{achieved}}, \rho_{\min} + (\rho_{\max} - \rho_{\min}) \cdot \Upsilon(\psi)), & \text{otherwise} \end{cases} \quad (4.7)$$

This formulation ensures that once the agent reaches a higher difficulty level, the curriculum will not regress, promoting stable and forward-moving learning dynamics.

- **Predefined Curriculum (Time-Based Logarithmic Schedule)**

In the predefined strategy which is fixed, time-based schedules, the range factor is not based on performance, but rather on training progress. Let the normalized training progress be

$$\tau = \frac{t}{T \cdot t_{\text{sat}}}, \quad 0 \leq \tau \leq 1,$$

where t is the current time-step, T is the total number of training steps, and t_{sat} is an optional “*saturation*” constant that can slow the clock. In the intermediate

4. Methodology

phase, we set the range factor $\rho(\tau)$ with a log-based approach that starts at $\rho_{\min}$ and reaches $\rho_{\max}$ at the end of training. The sampling range factor is then defined as:

$$\rho(\tau) = \begin{cases} \rho_{\min}, & \tau = 0, \\ \rho_{\max} \cdot \frac{\log(1+\tau)}{\log(2)}, & 0 < \tau < 1, \\ \rho_{\max}, & \tau = 1. \end{cases} \quad (4.8)$$

This strategy allows the agent to gradually experience more difficult tasks over time.

4.2 Dual Buffer Mechanism (DB)

Humans acquire skills through the reinforcement of successful experiences and the recollection of past failures to avoid their repetition. Building on this cognitive process, our proposed *Dual Buffer Mechanism (DB)*, a curriculum-based data-exploitation scheme, organizes replayed experiences into two specialized buffers—one for positive outcomes and another for negative ones. This structure enables the agent to concurrently consolidate effective behaviors and mitigate the recurrence of sub-optimal actions, thereby facilitating more robust and balanced policy learning.

The DB mechanism operates by categorizing episodes based on their total return R , and storing the corresponding transitions into one of three buffers: the positive buffer $\mathcal{B}_{\text{pos}}$, the negative buffer $\mathcal{B}_{\text{neg}}$, or the primary buffer $\mathcal{B}_{\text{ser}}$. Transitions are assigned according to the following criteria:

- $\mathcal{B}_{\text{pos}}$: contains transitions from episodes that succeed and satisfy $R \geq \vartheta_{\text{pos}}$.
- $\mathcal{B}_{\text{neg}}$: contains transitions from episodes that fail and satisfy $R \leq \vartheta_{\text{neg}}$.

To keep the thresholds ϑ_{pos} and ϑ_{neg} meaningful throughout training, we used different update strategies. The positive threshold ϑ_{pos} is scheduled using a time-driven function of normalized training progress, while the negative threshold ϑ_{neg} is held fixed. We define normalized training progress as:

$$k = \frac{t}{T}, \quad \text{where } 0 \leq k \leq 1$$

where t is the current timestep and T is the total number of training steps.

4. Methodology

Predefined Threshold for ϑ_{pos}

- **Linear:** In the linear scheduling approach, ϑ_{pos} increases at a constant rate from its initial value $\vartheta_{\text{pos}}^{\min}$ to its final value $\vartheta_{\text{pos}}^{\max}$, based on the normalized progress k :

$$\vartheta_{\text{pos}}(k) = \vartheta_{\text{pos}}^{\min} + k \cdot (\vartheta_{\text{pos}}^{\max} - \vartheta_{\text{pos}}^{\min}). \quad (4.9)$$

- **Exponential:** Alternatively, under the exponential profile, the threshold, ϑ_{pos} increases gradually at the beginning and accelerates toward the later phase of training:

$$\vartheta_{\text{pos}}(k) = \vartheta_{\text{pos}}^{\min} \cdot \left(\frac{\vartheta_{\text{pos}}^{\max}}{\vartheta_{\text{pos}}^{\min}} \right)^k. \quad (4.10)$$

Fixed Threshold for ϑ_{neg}

Unlike ϑ_{pos} , the threshold ϑ_{neg} for negative episodes remains fixed throughout training, defined as:

$$\vartheta_{\text{neg}} = \vartheta_{\text{neg}}^{\text{fix}}. \quad (4.11)$$

By scheduling ϑ_{pos} based solely on elapsed training time k , and not on performance metrics such as success rate, we ensure that every training run follows the threshold of accepting experiences in the positive buffer. This structure enhances experimental reproducibility and avoids instability due to noisy or fluctuating performance-based criteria.

Proportional Sampling Ratio

In reinforcement learning, particularly in sparse-reward environments, the strategy for sampling experiences from replay buffers plays a crucial role in shaping the learning dynamics. Our framework introduces a structured sampling mechanism that leverages three distinct buffers: the primary replay buffer $\mathcal{B}_{\text{ser}}$, the positive buffer $\mathcal{B}_{\text{pos}}$, and the negative buffer $\mathcal{B}_{\text{neg}}$. Each buffer contributes to training batches according to predefined or adaptive sampling ratios: ζ_{ser} (primary sampling ratio), ζ_{pos} (positive sampling ratio), and ζ_{neg} (negative sampling ratio), where

$$\zeta_{\text{ser}} + \zeta_{\text{pos}} + \zeta_{\text{neg}} = 1, \quad \text{where } \zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}} \in [0, 1]. \quad (4.12)$$

4. Methodology

These ratios determine how much data is drawn from each buffer during each training update. We explore two main strategies for scheduling these ratios: *Fixed Sampling* and *Adaptive Sampling*.

- **Fixed Sampling Ratios** In the fixed setup, each experience buffer contributes a constant proportion of samples throughout training. The sampling ratios $\zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}}$ are predefined constants that satisfy Eq. 4.12. This configuration ensures consistent sampling from the primary, positive, and negative buffers across all training phases, independent of the agent’s performance or learning progress.

- **Adaptive Sampling Strategies**

In curriculum-based experience replay, the sampling strategy is explicitly conditioned on the agent’s success rate $\psi \in [0, 1]$, enabling adaptive modulation of experience selection as learning progresses. We propose two adaptive sampling strategies: *Fully Adaptive* and *Hybrid Adaptive*.

- **Fully Adaptive Strategy:** All three sampling ratios $\zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}}$ are adjusted dynamically as functions of ψ :

$$\zeta_{\text{ser}}(\psi) = 1 - \zeta_{\text{pos}}(\psi) - \zeta_{\text{neg}}(\psi). \quad (4.13)$$

$$\zeta_{\text{pos}}(\psi) = \zeta_{\min} + (\zeta_{\max} - \zeta_{\min}) \cdot \psi, \quad (4.14)$$

$$\zeta_{\text{neg}}(\psi) = \zeta_{\min} + (\zeta_{\max} - \zeta_{\min}) \cdot (1 - \psi), \quad (4.15)$$

where $\zeta_{\min}, \zeta_{\max} \in (0, 1)$ are predefined bounds. This setup ensures that early in training ($\psi \rightarrow 0$), negative experiences are emphasized to guide learning. As the agent improves ($\psi \rightarrow 1$), sampling shifts toward positive experiences to strengthen successful behaviors.

- **Hybrid Adaptive Sampling Strategies:** In Hybrid Adaptive Sampling, one of the sampling ratios— $\zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}}$ is held fixed, while the remaining two are adaptively adjusted based on the agent’s success rate $\psi \in [0, 1]$. A common adaptive mixing coefficient $\zeta(\psi)$ is used to interpolate the proportions of the non-fixed buffers. The general linear interpolation form is:

$$\zeta(\psi) = \zeta_{\max} - (\zeta_{\max} - \zeta_{\min}) \cdot \psi, \quad (4.16)$$

4. Methodology

where $\zeta_{\min}, \zeta_{\max} \in (0, 1)$ are predefined bounds. In this process, one buffer $q \in \{\text{ser}, \text{pos}, \text{neg}\}$ is held at a constant weight ζ_f^{fix} . Of the remaining two buffers, one $h \in \{\text{ser}, \text{pos}, \text{neg}\} \setminus \{q\}$ —receives a proportion proportional to $\zeta(\psi)$, and the other receives the complement. Formally, for each $i \in \{\text{ser}, \text{pos}, \text{neg}\}$:

$$\zeta_i(s; q, h) = \begin{cases} \zeta_f^{\text{fix}}, & i = q, \\ (1 - \zeta_f^{\text{fix}}) \zeta(\psi), & i = h, \\ (1 - \zeta_f^{\text{fix}}) (1 - \zeta(\psi)), & \text{otherwise.} \end{cases} \quad (4.17)$$

By selecting (q, h) appropriately this unified expression recovers each of the six hybrid-buffer strategies. These hybrid strategies ensure that the curriculum of sampled experiences evolves smoothly, aligning with the agent’s competence while preserving a balanced diversity in the replay content.

The overall DB mechanism is summarized in Alg. 1, where episodes are sorted based on return and stored in the appropriate buffer. During training, samples are drawn according to either fixed or adaptive ratios to reinforce successful outcomes and avoid repeated failures.

4. Methodology

Algorithm 1 Dual Buffer Mechanism (DB)

```

1: Initialize: thresholds  $\vartheta_{\text{pos}}, \vartheta_{\text{neg}}$ , buffers  $\mathcal{B}_{\text{ser}}, \mathcal{B}_{\text{pos}}, \mathcal{B}_{\text{neg}}$   $\triangleright$  Setup DB thresholds and buffers
2: Initialize agent: policy  $\phi$ , critics  $\theta_1, \theta_2$   $\triangleright$  Agent initialization
3: while not converged do
4:   Select action  $a \sim \pi_\phi(s, g)$   $\triangleright$  Data Collection
5:   Step:  $s', r, d \leftarrow \text{Env.step}(a)$ , store  $(s, a, r, s', d) \in \mathcal{E}$ , set  $s \leftarrow s'$ 
6:   if episode terminates then
7:     Compute return  $R = \sum r_t$   $\triangleright$  Episode total reward
8:     if penalty applied and Dual Buffer enabled and  $R \leq \vartheta_{\text{neg}}$  then
9:       Store  $\mathcal{E} \rightarrow \mathcal{B}_{\text{neg}}$   $\triangleright$  Store failed episode in negative buffer
10:    else
11:      Store  $\mathcal{E} \rightarrow \mathcal{B}_{\text{ser}}$   $\triangleright$  Store in primary buffer (PER applies if enabled)
12:      if Dual Buffer enabled and  $R \geq \vartheta_{\text{pos}}$  then
13:        Store  $\mathcal{E} \rightarrow \mathcal{B}_{\text{pos}}$   $\triangleright$  Store successful episode in positive buffer
14:      end if
15:      if HER enabled then  $\triangleright$  HER (Optional)
16:        Relabel transitions  $\mathcal{E}_{\text{HER}} \leftarrow \text{HER}(\mathcal{E})$ , store in  $\mathcal{B}_{\text{ser}}$ 
17:      end if
18:    end if
19:    Update thresholds  $\vartheta_{\text{pos}}, \vartheta_{\text{neg}}$ , reset  $\mathcal{E} \leftarrow \emptyset$   $\triangleright$  Reset episode and update thresholds
20:  end if
21:  if Network update scheduled then
22:    if Dual Buffer enabled then
23:      Compute sampling ratios  $\zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}}$   $\triangleright$  Proportional sampling ratios
24:      Sample:  $\mathcal{D}_{\text{ser}} \sim \mathcal{B}_{\text{ser}}, \mathcal{D}_{\text{pos}} \sim \mathcal{B}_{\text{pos}}, \mathcal{D}_{\text{neg}} \sim \mathcal{B}_{\text{neg}}$   $\triangleright$  Sample from buffers
25:      Merge:  $\mathcal{D} = \mathcal{D}_{\text{ser}} \cup \mathcal{D}_{\text{pos}} \cup \mathcal{D}_{\text{neg}}$   $\triangleright$  Combine all batches
26:    else
27:      Sample batch:  $\mathcal{D} \sim \mathcal{B}_{\text{ser}}$   $\triangleright$  PER applies if enabled
28:    end if
29:    Update policy and critics:  $\phi, \theta_1, \theta_2 \leftarrow \text{Agent}(\mathcal{D})$   $\triangleright$  Policy update
30:    Update priorities in  $\mathcal{B}_{\text{ser}}$   $\triangleright$  Update priorities (PER applies if enabled)
31:    if Dual Buffer enabled then
32:      Update priorities in  $\mathcal{B}_{\text{pos}}, \mathcal{B}_{\text{neg}}, \mathcal{B}_{\text{ser}}$   $\triangleright$  Update DB priorities
33:    end if
34:  end if
35:  if evaluation scheduled then
36:    Evaluate  $\pi_\phi$  on fixed environment  $\triangleright$  Evaluation
37:     $R^{\text{eval}} \leftarrow R^{\text{eval}} + \left[ \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{|r_i|} \sum_{j=1}^{|r_i|} r_{i,j} \right) \right]$   $\triangleright$  Append mean reward
38:     $\psi^{\text{eval}} \leftarrow \psi^{\text{eval}} + \left[ \frac{1}{N} \sum_{i=1}^N \mathbb{1}_{\{\text{success}(g_i, \tau_i)\}} \right]$   $\triangleright$  Append mean success rate
39:     $L^{\text{eval}} \leftarrow L^{\text{eval}} + \left[ \frac{1}{N} \sum_{i=1}^N |t_i| \right]$   $\triangleright$  Append mean episode length
40:  end if
41: end while

```

4. Methodology

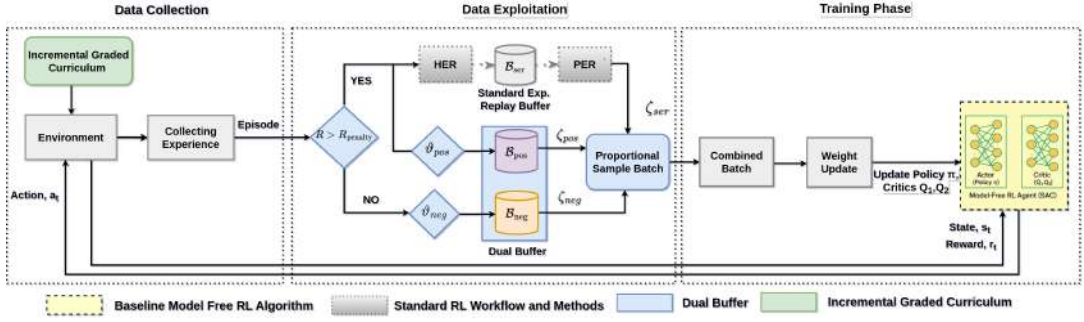


Figure 4.2: A schematic of the Dual-Mechanism Incremental Curriculum Learning (DICL) framework, where the baseline (model-free) algorithm is shown in **yellow**, the standard RL workflow and methods (HER, PER) in **gray**, the Dual-Buffer mechanism (DB) in **blue**, and the Incremental Goal Curriculum (IGC) module in **green**. For a detailed description, see Alg. 2.

4.3 Dual-Mechanism Incremental CL Framework

We propose a unified architecture titled the *Dual-Mechanism Incremental Curriculum Learning (DICL)* framework shown in Alg. 2 that integrates curriculum-driven data collection and dual-buffer-driven data exploitation to enhance the learning process of sparse-reward robotic tasks.

As shown in Fig. 4.2, during training, the agent receives the current state s_t from the environment and selects an action a_t according to its policy π . The environment then executes this action and returns the next state s_{t+1} , and the reward r_t . The transition tuple (s_t, a_t, r_t, s_{t+1}) is stored as part of an episode.

The data collection process is guided by the *IGC*, which gradually increases task complexity based on either the elapsed training time or the agent’s ongoing performance. This curriculum provides a structured exploration strategy, enabling the agent to progressively learn from simpler to more complex scenarios.

Each episode generates a sequence of transitions $\{(s_t, a_t, r_t, s_{t+1})\}_{t=1}^T$, where each individual reward r_t may incorporate a penalty term $\mathcal{R}_{\text{penalty}}(t)$ if a penalizing event occurs at time step t . If a penalty event occurs during the episode and the failure episode return further satisfies $R \leq v_{\text{neg}}$, where v_{neg} is a fixed threshold, then all transitions from the episode are stored in the negative buffer $\mathcal{B}_{\text{neg}}$.

Conversely, if no penalty event is triggered, the full sequence of transitions from the episode is added to the standard replay buffer, $\mathcal{B}_{\text{ser}}$. Furthermore, if the episode is successful and the total return satisfies the condition $R \geq v_{\text{pos}}$, all transitions from

4. Methodology

the episode are stored in the positive buffer $\mathcal{B}_{\text{pos}}$. The threshold ϑ_{pos} is predefined and progressively increases over the course of training, either linearly or exponentially, depending on the selected schedule. Upon successful task completion, the episode is immediately terminated at that timestep, and no further environment interactions are recorded.

To further enhance sample efficiency, the proposed framework optionally integrates HER and PER. These mechanisms can be activated individually or jointly, depending on the characteristics and complexity of the task environment.

During the data exploitation phase, transitions samples are drawn from three distinct buffers— $\mathcal{B}_{\text{ser}}$, $\mathcal{B}_{\text{pos}}$, and $\mathcal{B}_{\text{neg}}$ —according to predefined sampling ratios ζ_{ser} , ζ_{pos} , and ζ_{neg} , corresponding to the standard replay buffer, the positive buffer, and the negative buffer, respectively. These ratios may be defined statically or adjusted dynamically via an adaptive mechanism, depending on the progression and objectives of the curriculum.

In the training phase, transitions are sampled from each buffer using the selected ratios and aggregated into a unified *Combined Batch*. Once a sufficient amount of experience has been accumulated, this batch is employed to perform gradient-based updates on the SAC agent. Specifically, both the policy network π and the twin critic networks Q_1 and Q_2 are optimized through their respective loss functions. These weight updates are executed periodically and remain decoupled from the real-time interaction loop with the environment.

4. Methodology

Algorithm 2 Dual-Mechanism Incremental Curriculum Learning (DICL)

```

1: Initialize: curriculum progression coeff  $n \leftarrow 0$ , thresholds  $\vartheta_{\text{pos}}, \vartheta_{\text{neg}}$ , buffers
    $\mathcal{B}_{\text{ser}}, \mathcal{B}_{\text{pos}}, \mathcal{B}_{\text{neg}}$ 
2: Initialize agent: policy  $\phi$ , critics  $\theta_1, \theta_2$  ▷ Agent setup
3: Set environment state  $s \leftarrow r_{\text{obj}}(n)$ ,  $g \leftarrow r_{\text{goal}}(n)$  ▷ Initialize IGC
4: while not converged do
5:   Select action  $a \sim \pi_{\phi}(s, g)$  ▷ Data Collection
6:   Step:  $s', r, d \leftarrow \text{Env.step}(a)$ , store  $(s, a, r, s', d) \in \mathcal{E}$ , set  $s \leftarrow s'$ 
7:   if episode terminates then
8:     Compute return  $R = \sum r_t$  ▷ Calculate total episode reward
9:     if penalty applied and Dual Buffer enabled and  $R \leq \vartheta_{\text{neg}}$  then
10:       Store  $\mathcal{E} \rightarrow \mathcal{B}_{\text{neg}}$  ▷ Store failed episode in negative buffer
11:     else
12:       Store  $\mathcal{E} \rightarrow \mathcal{B}_{\text{ser}}$  ▷ Store in primary replay buffer (PER applies if enabled)
13:       if Dual Buffer enabled and  $R \geq \vartheta_{\text{pos}}$  then
14:         Store  $\mathcal{E} \rightarrow \mathcal{B}_{\text{pos}}$  ▷ Store successful episode in positive buffer
15:       end if
16:       if HER enabled then ▷ HER (Optional)
17:         Relabel transitions  $\mathcal{E}_{\text{HER}} \leftarrow \text{HER}(\mathcal{E})$ , store in  $\mathcal{B}_{\text{ser}}$ 
18:       end if
19:     end if
20:     Update thresholds  $\vartheta_{\text{pos}}, \vartheta_{\text{neg}}$ , reset  $\mathcal{E} \leftarrow \emptyset$  ▷ Reset episode and update thresholds
21:     Update  $n$ , set  $s \leftarrow r_{\text{obj}}(n)$ ,  $g \leftarrow r_{\text{goal}}(n)$  ▷ Update IGC
22:   end if
23:   if Network update scheduled then
24:     if Dual Buffer enabled then
25:       Compute sampling ratios  $\zeta_{\text{ser}}, \zeta_{\text{pos}}, \zeta_{\text{neg}}$  ▷ Proportional sampling ratios
26:       Sample:  $\mathcal{D}_{\text{ser}} \sim \mathcal{B}_{\text{ser}}, \mathcal{D}_{\text{pos}} \sim \mathcal{B}_{\text{pos}}, \mathcal{D}_{\text{neg}} \sim \mathcal{B}_{\text{neg}}$  ▷ Sample from buffers
27:       Merge:  $\mathcal{D} = \mathcal{D}_{\text{ser}} \cup \mathcal{D}_{\text{pos}} \cup \mathcal{D}_{\text{neg}}$  ▷ Combine all batches
28:     else
29:       Sample batch:  $\mathcal{D} \sim \mathcal{B}_{\text{ser}}$  ▷ PER applies if enabled
30:     end if
31:     Update policy and critics:  $\phi, \theta_1, \theta_2 \leftarrow \text{Agent}(\mathcal{D})$  ▷ Update policy and critics
32:     Update priorities in  $\mathcal{B}_{\text{ser}}$  ▷ Update priorities (PER applies if enabled)
33:     if Dual Buffer enabled then
34:       Update priorities in  $\mathcal{B}_{\text{pos}}, \mathcal{B}_{\text{neg}}, \mathcal{B}_{\text{ser}}$  ▷ Update DB priorities
35:     end if
36:   end if
37:   if evaluation scheduled then
38:     Evaluate  $\pi_{\phi}$  with  $s \leftarrow r_{\text{obj}}(n=1)$ ,  $g \leftarrow r_{\text{goal}}(n=1)$  ▷ Evaluate agent
39:      $R^{\text{eval}} \leftarrow R^{\text{eval}} + \left[ \frac{1}{N} \sum_{i=1}^N \left( \frac{1}{|r_i|} \sum_{j=1}^{|r_i|} r_{i,j} \right) \right]$  ▷ Append mean reward
40:      $\psi^{\text{eval}} \leftarrow \psi^{\text{eval}} + \left[ \frac{1}{N} \sum_{i=1}^N \mathbb{1}_{\{\text{success}(g_i, \tau_i)\}} \right]$  ▷ Append mean success rate
41:      $L^{\text{eval}} \leftarrow L^{\text{eval}} + \left[ \frac{1}{N} \sum_{i=1}^N |t_i| \right]$  ▷ Append mean episode length
42:   end if
43: end while

```

Chapter 5

Result Analysis

This chapter presents a comprehensive empirical evaluation across three sparse-reward robotic tasks, systematically comparing standard baselines with our proposed approaches: IGC, DB, and their integration within DICL framework. We analyze a range of evaluation metrics to quantify the improvements in sample efficiency and training stability introduced by these CL strategies. Furthermore, a comparative assessment is conducted to identify the most effective method for each task.

5.1 Experimental Setup

We evaluate our approach in various sparse-reward tasks of the Gymnasium-Robotics Fetch benchmark environment [55] such as *FetchPush* (*Pushing* task), *FetchPickAndPlace* (*Pick-and-Place* task), and *FetchSlide* (*Sliding* task) shown in Fig. 5.1. These environments follow a multi-goal RL setting, where each episode involves achieving a different goal state, and the agent is trained to generalize across these varying goals. We benchmark our proposed methods—IGC, DB, and their integration in DICL—compared to baseline agents to evaluate their effectiveness.

5.2 Failure States

To prevent undesired behaviors such as dropping the object off the table, we explicitly define a class of *failure states*, denoted by s_{fail} . In this context, a failure state corresponds to any situation where the object’s height falls below a predefined safety threshold, indicating that it has fallen off the table.

5. Result Analysis

To penalize such failure states, we introduce a drop penalty defined as:

$$R_{\text{penalty}}(t) = \begin{cases} -100, & \text{if } z_{\text{obj}}(t) < z_{\text{thres}} \\ 0, & \text{otherwise} \end{cases} \quad (5.1)$$

Here, $z_{\text{obj}}(t)$ denotes the vertical position (z-coordinate) of the object at time t , and z_{thres} represents the threshold height below which the state is considered a failure. When $R_{\text{penalty}}(t) = -100$, it signals the agent has entered a failure state s_{fail} , typically due to the object falling off the table.

5.3 Training and Evaluation Protocol

Our baseline (SAC) algorithm builds upon using OpenAI’s SpinningUp framework [56], described in Appendix B, with custom modifications of hyperparameters shown in Tab. 5.1.

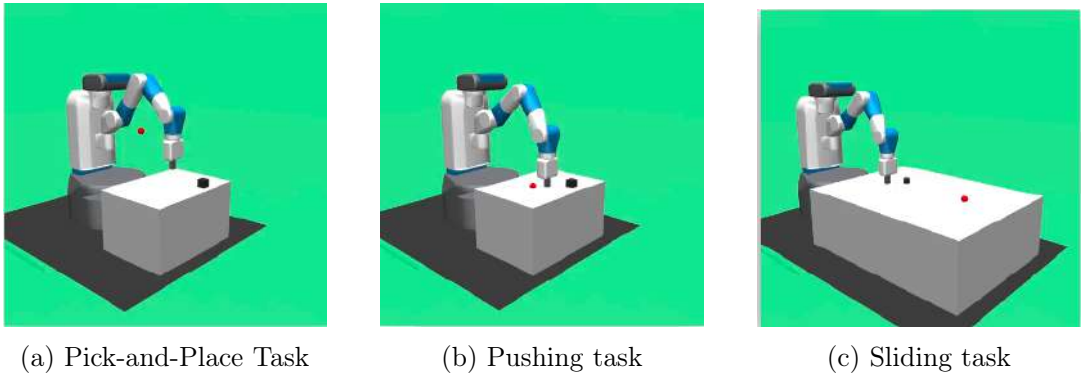


Figure 5.1: Gymnasium-Robotics Fetch Benchmark Environment Tasks

Our evaluation is structured around three main configurations to systematically analyze the contributions of the proposed components.

- A baseline where the agent is trained using SAC combined solely with the IGC, without any drop-penalty or DB mechanism.
- A baseline where SAC is augmented with drop-penalty reward shaping and the DB mechanism, but without IGC. We also include HiER [49], a state-of-the-art CL strategy, as a strong baseline under this configuration.
- Our full proposed DICL pipeline, integrating both IGC and DB, evaluated against the SAC baseline with drop-penalty to isolate the benefit of CL.

5. Result Analysis

Each of these configurations is additionally evaluated with HER, PER, and their combination to assess compatibility and cumulative performance gains. All experiments are run for 500K timesteps and repeated with 3 different random seeds.

We evaluate agent performance at fixed intervals using 100 deterministic episodes (with no exploration noise). The *success rate* is defined as the fraction of episodes in which the agent successfully achieves the goal, computed as $\frac{1}{100} \sum_{i=1}^{100} \mathbb{1}\{\text{success}_i\}$. The *cumulative reward* is the average total reward accumulated per episode, given by $\frac{1}{100} \sum_{i=1}^{100} \sum_{t=1}^{T_i} r_t^{(i)}$, where T_i is the number of steps in episode i . The *episode length* is the average number of steps per episode, computed as $\frac{1}{100} \sum_{i=1}^{100} T_i$. These metrics are recorded periodically in every 500 timesteps to monitor learning progress and evaluate policy performance.

Table 5.1: The Hyperparameters of Baseline SAC

Hyperparameter	Value
Discount factor, γ	0.95
Entropy coefficient, α	0.1
Learning rate	0.001
Actor & critic architecture	3 hidden layers of 256 units
Activation function	ReLU
Replay buffer size	10^6
Batch size	256
Polyak averaging coefficient, τ	0.995
Action squashing	Tanh
Boundary min	-1.0
Boundary max	1.0
Exploration (random) steps	First 1 000 steps
Max episode length	50
Total_Timesteps	500 000
Number of test episodes	100
Update interval	50 (training update every 50 steps)
Eval freq	500 (evaluate every 500 steps)
PER exponent, α_{PER}	0.6
PER initial β , β_{start}	0.4
HER strategy	Future
HER transitions/episode	4 goal transitions per episode

5.4 Evaluation Metrics

To evaluate the overall performance across all tasks, we compare our proposed methods with existing state-of-the-art baselines using four key metrics: *Mean*, *Standard Deviation (StdDev)*, *Median*, and *Optimality Gap (OG)*. Each metric is

5. Result Analysis

computed from the best per-task performance across multiple seeds to ensure a stable and fair comparison. The Mean reflects the average success rate over multiple seeds, while the StdDev captures the variability in performance, with lower values indicating more stable learning. The Median provides robustness to outliers by capturing the central value. The OG quantifies how far a method is from the ideal performance (e.g., success rate of 1.0), and is computed as $OG = 1 - \text{Mean}$. Although the OG is mathematically linked to the Mean in this setting, we report both for completeness and to emphasize how closely each method approaches optimal task success.

Tab. 5.2 provides a comparative analysis of our proposed methods and state-of-the-art baselines across three robotic tasks using four key evaluation metrics: *Mean*, *Standard Deviation (StdDev)*, *Median*, and *Optimality Gap (OG)*. As shown in Fig. 5.2, each horizontal bar represents the 95% confidence interval (CI) of the success rate across three seeds for a given method, while the vertical black line within each bar indicates the corresponding point estimate (e.g., mean, median, or OG). These confidence intervals were computed using bootstrap resampling—a non-parametric statistical technique [57] that generates multiple synthetic datasets by sampling with replacement from the original set of seed-level results. This method enables robust estimation of uncertainty, particularly in scenarios with limited trials. Narrower CIs imply more consistent performance across seeds, while wider intervals reflect higher variability.

In the Pick-and-Place task, our methods **IGC+HER+PER [Fully Adaptive]**, **DB+HER [Fixed ζ]**, and **DICL+HER** achieve perfect performance, each with a Mean of 1.00, Median of 1.00, StdDev of 0.00, and OG of 0.00. In contrast, the state-of-the-art method **HiER+HER** achieves a lower Mean of 0.76, Median of 0.76, and a higher StdDev of 0.19 and OG of 0.24. Other baselines, such as **Baseline+HER** and **Baseline+HER+PER**, yield Mean success rates of only 0.44 and 0.37, respectively.

In the Pushing task, our methods again outperform the baselines. **IGC+HER+PER [Fully Adaptive]** and **DB+HER [Fixed ζ]** achieve perfect performance with a Mean and Median of 1.00, StdDev of 0.00, and OG of 0.00. **DICL+HER** also performs strongly, with a Mean of 0.97, StdDev of 0.06, and OG of 0.03. The strongest baseline, **HiER+HER**, achieves a Mean of 0.93, Median of 0.93, StdDev of 0.08, and OG of 0.07. All other baselines perform below 0.50 in mean success rate, demonstrating limited effectiveness in this task.

5. Result Analysis

Table 5.2: Our proposed method compared to the state-of-the-art methods based on success rate. The best results per task are highlighted in bold.

	Method	Pick-and-Place	Pushing	Sliding	Mean $\uparrow$	StdDev $\downarrow$	Median $\uparrow$	OG $\downarrow$
State-of-the-Art	Baseline+HER	✓			0.44	0.15	0.44	0.56
			✓		0.50	0.36	0.50	0.50
				✓	0.06	0.06	0.06	0.94
	Baseline+HER+PER	✓			0.37	0.15	0.37	0.63
			✓		0.10	0.12	0.10	0.90
				✓	0.06	0.12	0.06	0.94
	Baseline+Penalty+HER	✓			0.09	0.06	0.09	0.91
			✓		0.09	0.06	0.09	0.91
				✓	0.05	0.01	0.05	0.95
	Baseline+Penalty+PER	✓			0.10	0.00	0.10	0.90
			✓		0.10	0.12	0.10	0.90
				✓	0.05	0.00	0.05	0.95
Our Proposed	Baseline+Penalty+HER+PER	✓			0.09	0.05	0.09	0.910
			✓		0.10	0.06	0.10	0.90
				✓	0.06	0.00	0.06	0.94
	HiER+HER	✓			0.76	0.19	0.76	0.24
			✓		0.93	0.08	0.93	0.07
	HiER			✓	0.75	0.03	0.75	0.25
	IGC+HER+PER [Fully Adaptive]	✓			1.00	0.00	1.00	0.00
			✓		1.00	0.00	1.00	0.00
				✓	0.98	0.01	0.98	0.02
	IGC+HER [Fully Adaptive]	✓			1.00	0.00	1.00	0.00
			✓		0.99	0.03	0.99	0.01
				✓	0.75	0.01	0.75	0.25
	IGC+PER	✓			0.10	0.05	0.10	0.90
			✓		0.10	0.06	0.10	0.90
				✓	0.14	0.12	0.14	0.86
	DB+HER [Fixed ζ]	✓			1.00	0.00	1.00	0.00
			✓		1.00	0.00	1.00	0.00
				✓	0.73	0.06	0.73	0.27
	DB+HER [Hybrid Adaptive ζ]			✓	0.75	0.05	0.75	0.25
	DICL+HER	✓			1.00	0.00	1.00	0.00
			✓		0.97	0.06	0.97	0.03
				✓	0.80	0.12	0.80	0.20

5. Result Analysis

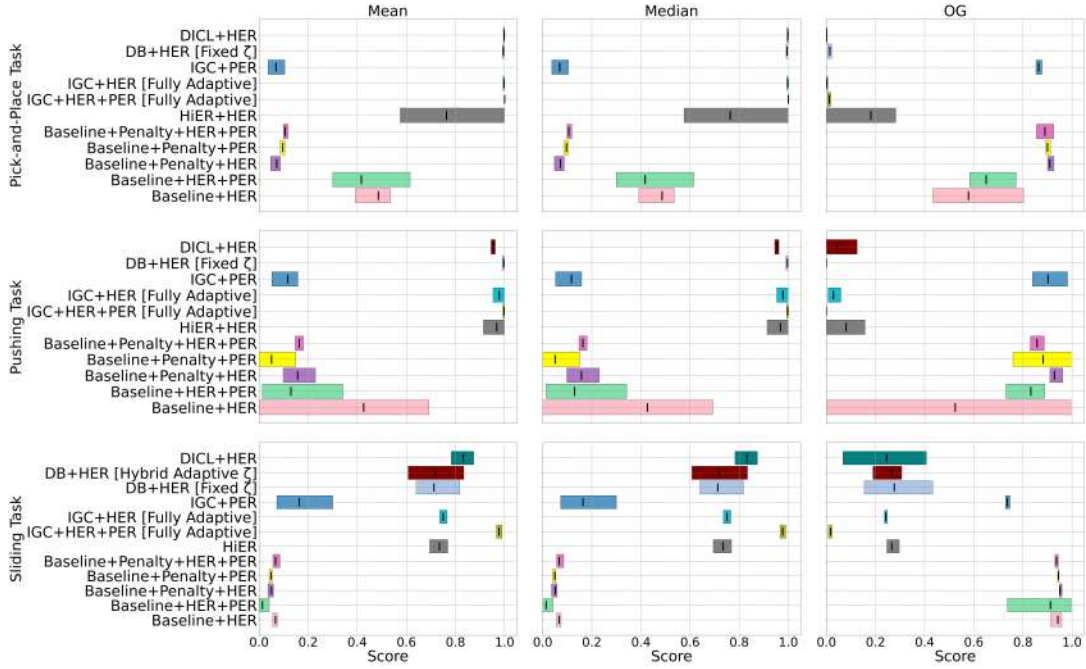


Figure 5.2: Comparison between our proposed methods and state-of-the-art baselines based on success rate with 95% confidence intervals. Point estimates are provided in Tab. 5.2.

In the Sliding task, which is the most challenging due to its unstable reward dynamics, IGC+HER+PER [Fully Adaptive] achieves the best result with a Mean of 0.98, Median of 0.98, StdDev of 0.01, and OG of 0.02. Other strong methods include DIACL+HER (Mean = 0.80, Median = 0.80, OG = 0.20) and DB+HER [Fixed ζ] (Mean = 0.73, Median = 0.73, OG = 0.27). Among the baselines, only HiER performs competitively with a Mean of 0.75, StdDev of 0.03, and OG of 0.25. All other variants perform below 0.15 in Mean success rate.

As shown by both tabular metrics and visual comparisons, baseline SAC agents—even when enhanced with HER or PER—struggle in sparse-reward environments. These agents frequently become stuck in local optima due to the lack of informative feedback, limiting their ability to explore effectively. While HER and PER improve sample efficiency, they remain insufficient to fully address the underlying exploration challenges.

In contrast, our curriculum-based methods—IGC, DB, and DIACL—progressively increase task difficulty, enabling structured skill acquisition, denser reward feedback, and faster convergence. Unlike HiER, which retains only successful episodes, our approach maintains both successful and unsuccessful experiences across varying difficulty levels. This inclusive replay strategy enhances training diversity and

reduces the risk of overfitting to a narrow subset of experiences. Overall, our proposed curriculum learning methods consistently outperform existing state-of-the-art approaches, achieving higher success rates, minimal performance variability, and significantly lower optimality gaps across all robotic applications.

5.5 Comparative Performance Evaluation of CL Strategies on Sparse-Reward Robotic Tasks

This section provides a rigorous empirical evaluation of the proposed methods—IGC, DB, and their unified variant DICL—across three benchmark sparse-reward robotic manipulation environments from the Gymnasium Fetch Robotics: *FetchPickAndPlace*, *FetchPush*, and *FetchSlide*. Performance is assessed using three performance metrics: evaluation success rate, cumulative reward, and episode length. All experiments are conducted over 500 K training timesteps, with evaluation carried out every 500 steps using 100 deterministic episodes per seed. Results are averaged across three independent training runs to account for stochasticity in learning.

The resulting performance curves depict the mean values across seeds, while shaded bands denote one standard deviation, thereby illustrating both central trends and variance across training runs. To facilitate a clear and task-specific interpretation, the analysis is structured per environment, beginning with *FetchPickAndPlace*, followed by *FetchPush*, and concluding with *FetchSlide*. Within each task, we systematically compare all proposed methods against established baselines based on performance metrics, highlighting convergence behavior, sample efficiency, and overall learning stability.

5.5.1 FetchPickAndPlace

As shown in Fig. 5.3, the evaluation success rate curve in Fig. 5.3a demonstrates that the Fully Adaptive strategy enables significantly faster convergence and more stable learning compared to the Self-Paced Adaptive and Predefined variants. The Fully Adaptive variant achieves higher success rates earlier, indicating enhanced task proficiency. In contrast, the Predefined curriculum lags behind, while the Self-Paced strategy performs moderately well but lacks the consistency and rapid improvement

5. Result Analysis

exhibited by the Fully Adaptive variant. We employ the Fully Adaptive IGC strategy in our Pick-and-Place task.

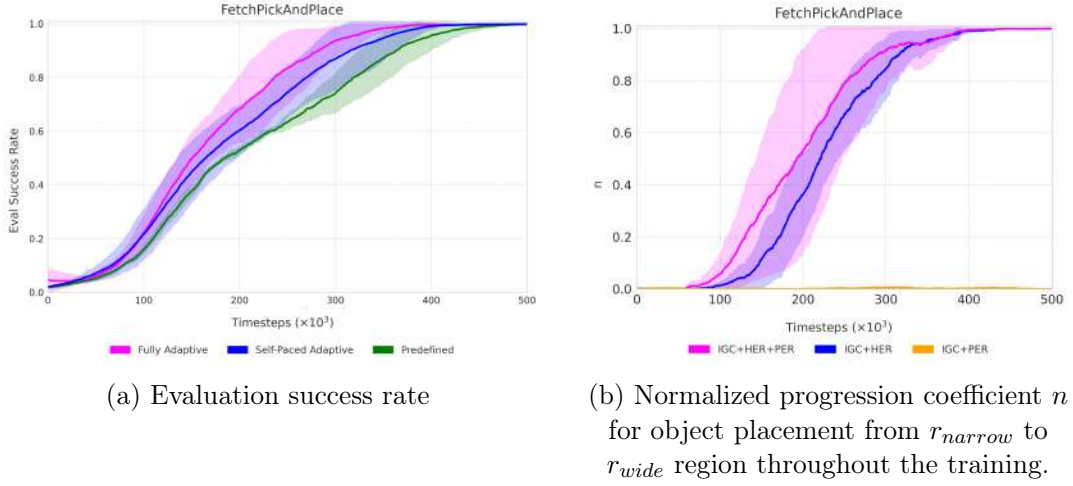


Figure 5.3: Evaluation success rate for Pick-and-Place task under three IGC variants and Progression coefficient n for the Fully Adaptive IGC during training.

The logarithmic increase in the progression coefficient n , shown in Fig. 5.3b, reflects a smooth curriculum transition from easy to hard tasks. A value of $n = 0$ defines the object as initialized inside a narrow region close to the target, yielding the easiest setting, while $n = 1$ represents that the object can be placed anywhere within the full workspace, corresponding to the hardest setting.

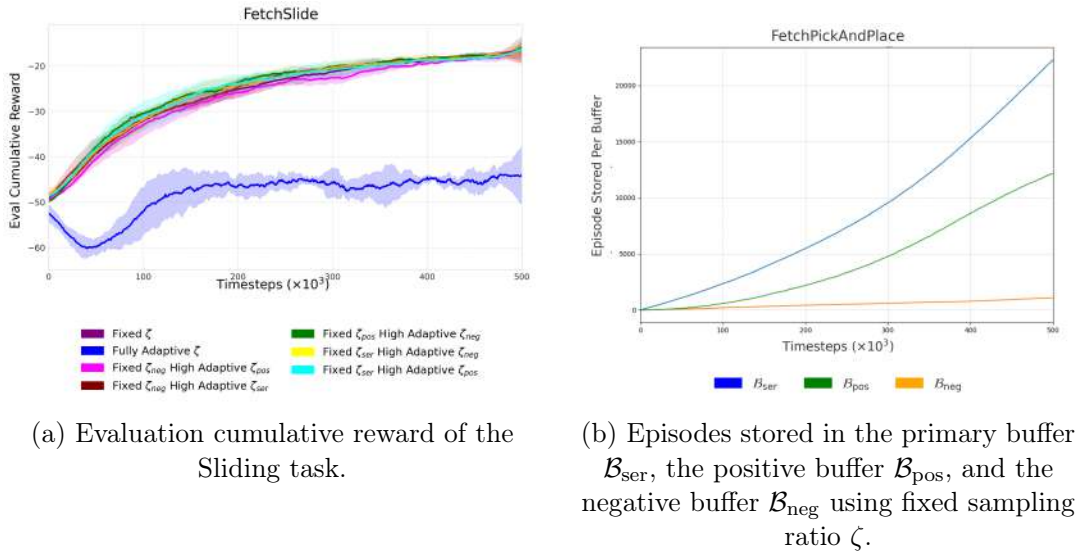
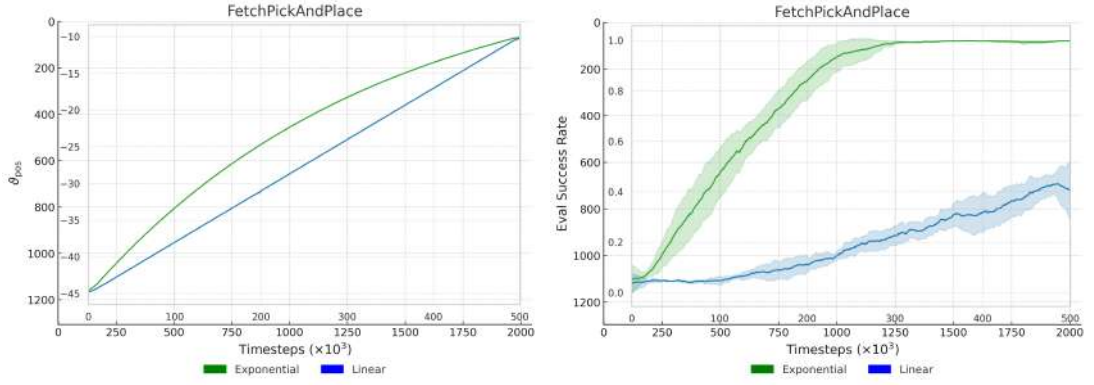


Figure 5.4: Comparison of Fixed and Hybrid Adaptive proportional-sampling strategies on the Sliding task and Cumulative number of episodes stored in the three replay buffers during training in the Pick-and-Place task.

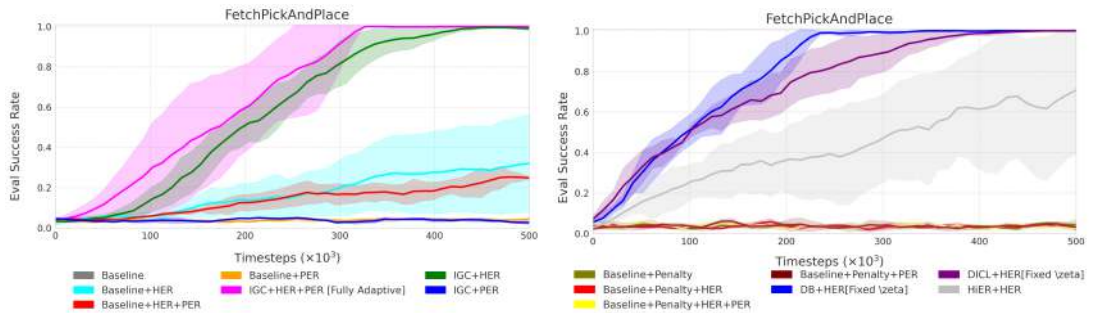
5. Result Analysis


 (a) Predefined threshold for v_{pos}

 (b) Evaluation success rate for v_{pos}

 Figure 5.5: v_{pos} changes over time and Evaluation performance between Exponential and Linear predefined thresholds for v_{pos}

Fig. 5.4b illustrates the cumulative number of episodes stored in each buffer over 500K timesteps. The primary buffer $\mathcal{B}_{\text{ser}}$ grows the fastest, reaching over 22,000 episodes, as it stores all collected experiences. In contrast, the positive buffer $\mathcal{B}_{\text{pos}}$ accumulates more gradually, reaching approximately 12,000 episodes, while the negative buffer $\mathcal{B}_{\text{neg}}$ remains relatively small, with only around 1,000 episodes stored. As shown in Fig. 5.5, we parameterize the positive threshold v_{pos} using both exponential and linear schedules, varying it smoothly from -45 at the beginning of training to -10 at the end. An exponential schedule for the predefined threshold v_{pos} yields superior performance compared to a linear schedule (see Fig. 5.5a). In our evaluations of DB and DIQL, we set the fixed sampling ratios to $\zeta_{\text{ser}} = \zeta_{\text{pos}} = \zeta_{\text{neg}} = 0.33$.



(a) Baselines compared to IGC

(b) Baselines Compared to DB and DIQL

Figure 5.6: Evaluation success rate of Pick-and-Place task compared to the Baseline methods for various agent configurations.

In Fig. 5.6a, the Fully Adaptive IGC+HER+PER variant demonstrates consistently superior performance, reaching near-optimal success rates more rapidly and with lower variance than both IGC+HER and all baseline methods. The narrower

5. Result Analysis

shaded region for IGC+HER+PER indicates higher training stability across three seeds. Fig. 5.6b shows that DB+HER with a fixed sampling ratio ζ achieves the fastest convergence—around 300K timesteps—outperforming both the fully adaptive and fixed-ratio variants of DIQL+HER, which converge near 429K steps.

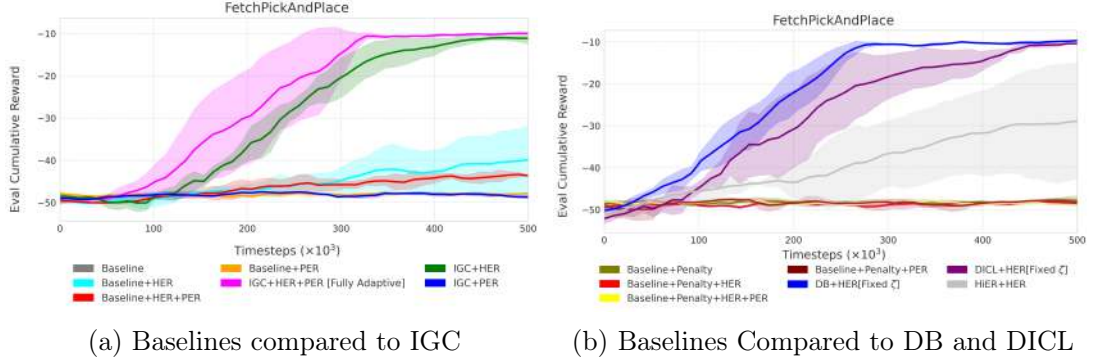


Figure 5.7: Cumulative reward of Pick-and-Place task compared to the Baseline methods for various agent configurations.

As illustrated in Fig. 5.7, DB+HER also reaches the highest cumulative reward (approximately -10) more quickly. Additionally, Fig. 5.8 confirms that it converges to the shortest episode length (10 steps), highlighting the method’s superior learning efficiency.

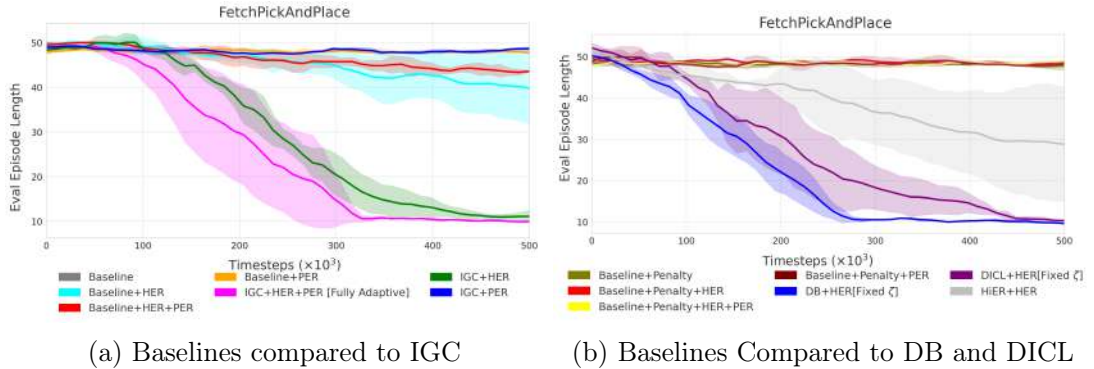


Figure 5.8: Episode Length of Pick-and-Place task compared to the Baseline methods for various agent configurations.

In the Pick-and-Place task, we employ an exponential schedule for the predefined threshold ϑ_{pos} to identify positive transitions, and apply a fixed proportional sampling strategy with a constant ratio of $\zeta = 0.33$. In all other tasks where fixed proportional sampling is used, this ratio of 0.33 is consistently maintained. The same scheduling and sampling configurations are applied across the additional tasks discussed in Subsections 5.5.2 and 5.5.3.

5. Result Analysis

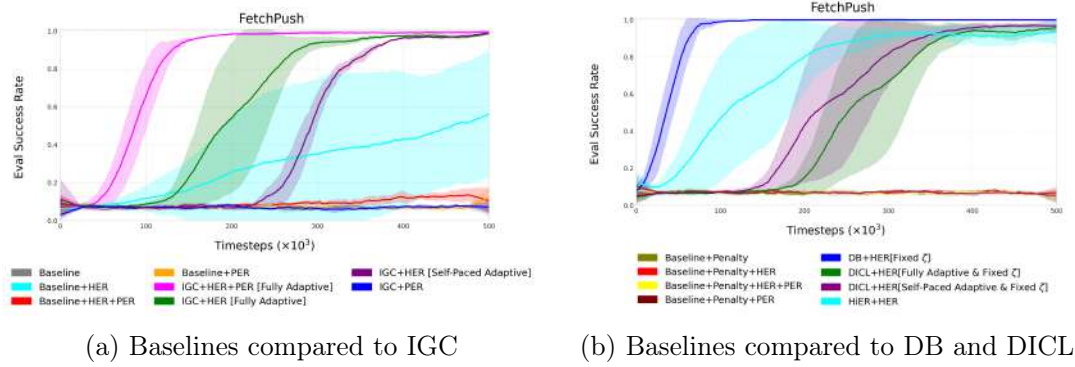


Figure 5.9: Evaluation Success Rate of Pushing task compared to the Baseline methods for various agent configurations.

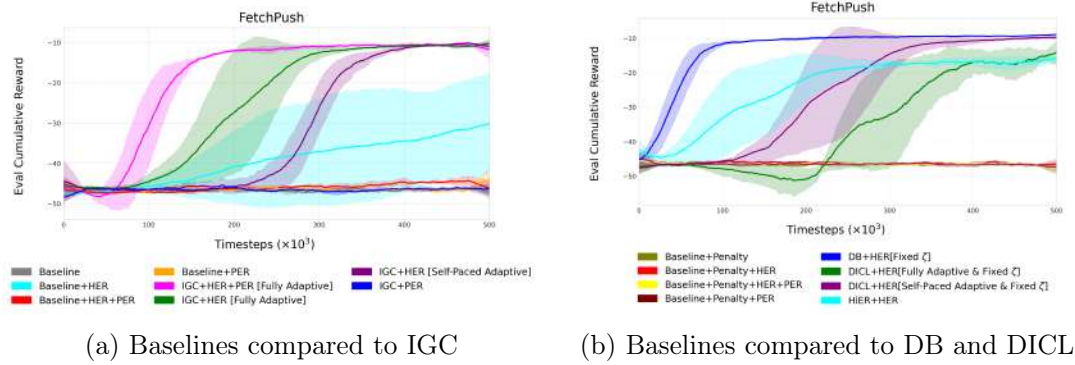


Figure 5.10: Cumulative reward of Pushing task compared to the Baseline methods for various agent configurations.

5.5.2 FetchPush

In Fig. 5.9b, the DB+HER variant with a fixed sampling ratio ζ converges to approximately 100% success by just 120K timesteps, outperforming all Dicl variants and baseline agents. Similarly, in Fig. 5.9a, the fully adaptive curriculum variant IGC+HER+PER reaches 100% success at around 282K timesteps. In contrast, all baseline methods—including those enhanced with HER or PER—remain below 10% success throughout training, highlighting their poor performance in sparse-reward settings. Further, as shown in Fig. 5.10, DB+HER not only achieves the highest cumulative return but does so earlier than all other configurations. It also converges to the shortest episode trajectories, as depicted in Fig. 5.11, indicating more efficient learning and better policy performance. These findings confirm that the DB+HER configuration offers the highest sample efficiency and learning stability on the Pushing task.

5. Result Analysis

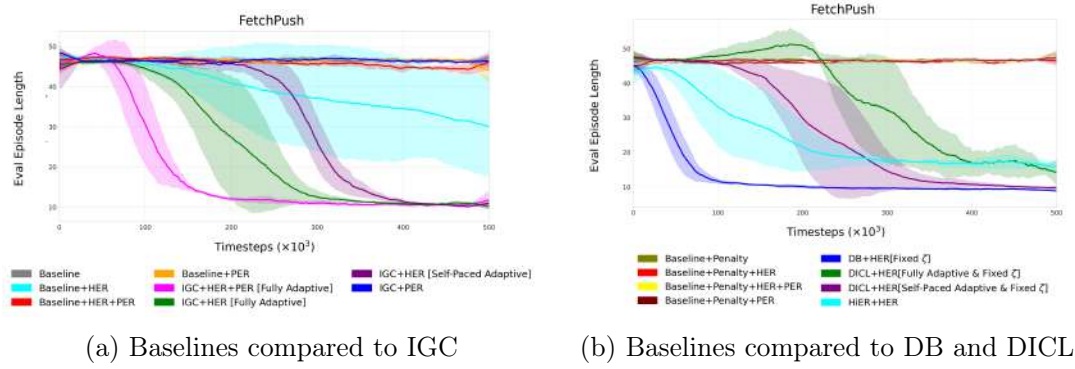


Figure 5.11: Evaluation episode length of Pushing task compared to the Baseline methods for various agent configurations.

5.5.3 FetchSlide

All six hybrid-adaptive sampling strategies introduced in Eq. 4.17 consistently outperform the fixed-ratio on the *FetchSlide* task in terms of early cumulative reward acquisition. As shown in Fig. 5.4a, they reach a cumulative reward of approximately -30 between 110K and 120K timesteps—about 10–20K steps earlier than the fixed strategy. Although all hybrid variants eventually converge around -18 cumulative reward, their early learning trajectories are consistently superior. Among all hybrid-adaptive strategies, the `Hybrid[Fixed Primary, High Adaptive Positive]` configuration (cyan) exhibits the fastest improvement in the early stages of training. However, by the end of training, all hybrid variants converge to similar levels of cumulative reward, indicating that their final performance is largely equivalent. Notably, the Fully Adaptive strategy (blue) consistently underperforms, exhibiting lower cumulative rewards throughout training, and is therefore excluded from further comparative analysis. These results suggest that hybrid buffer sampling, by combining fixed and adaptive prioritization, improves early learning efficiency and overall task performance in sparse-reward robotic environments.

As illustrated in Fig. 5.12–5.14, the hybrid adaptive buffer strategy employed in `DICL+HER` consistently demonstrates superior learning performance. Both the Fully Adaptive and Hybrid Sampling Ratio ζ configurations of `DICL+HER` attain success rates nearing 80% within 500K timesteps, while `HiER` and the hybrid buffer variant `DB+HER` converge around 75%, outperforming all baseline methods augmented with penalty rewards. Additionally, these `DICL`-based approaches exhibit greater learning efficiency, converging to an average cumulative reward of approximately -18 and

5. Result Analysis

stabilizing at episode lengths near 18 steps.

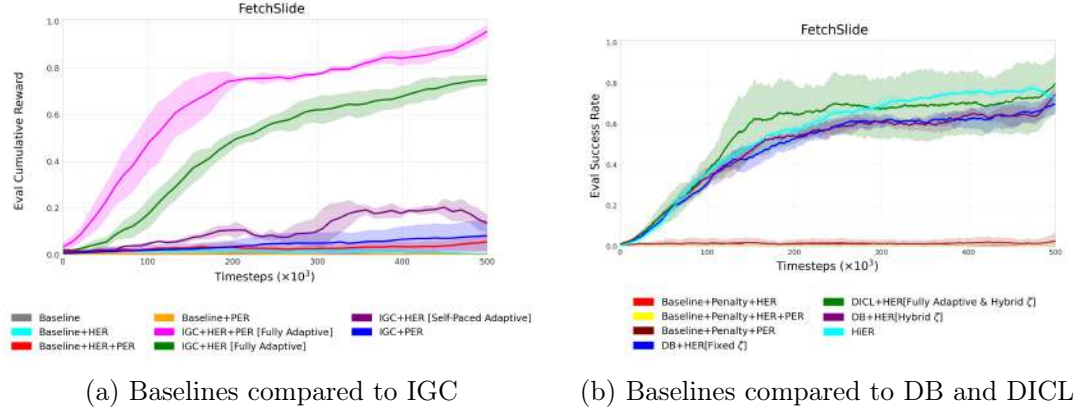


Figure 5.12: Evaluation success rate of Sliding task compared to the Baseline methods for various agent configurations.

Notably, the Fully Adaptive IGC+HER+PER variant emerges as the most effective strategy across all metrics. As shown in Fig. 5.12a and Fig. 5.13b, this method achieves a near-optimal success rate of 98%, the highest cumulative return reached, around -22 , and the shortest average episode length of approximately 22 steps (Fig. 5.14b).

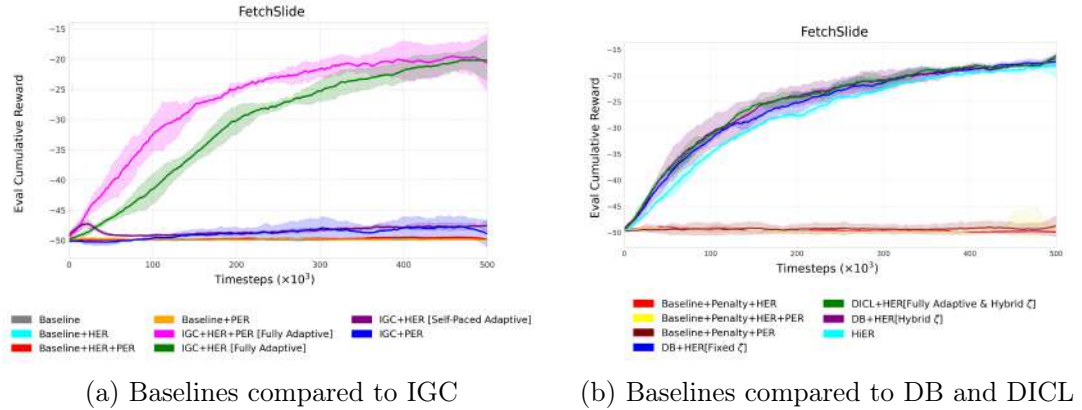


Figure 5.13: Evaluation cumulative reward of Sliding task compared to the Baseline methods for various agent configurations.

While DICL-based variants demonstrate marginal improvements in cumulative reward and trajectory length, their overall success rates remain comparatively lower. In contrast, the strength of the IGC+HER+PER strategy lies in its emphasis on *near-success* transitions—trajectory segments where the agent narrowly fails to achieve the goal. By prioritizing these transitions, the agent benefits from more targeted corrective learning, thereby improving its capacity to achieve consistent task success.

5. Result Analysis

Although this approach results in slightly longer episodes or lower average rewards, it significantly enhances reliability and robustness, particularly in sparse-reward settings or safety-critical applications.

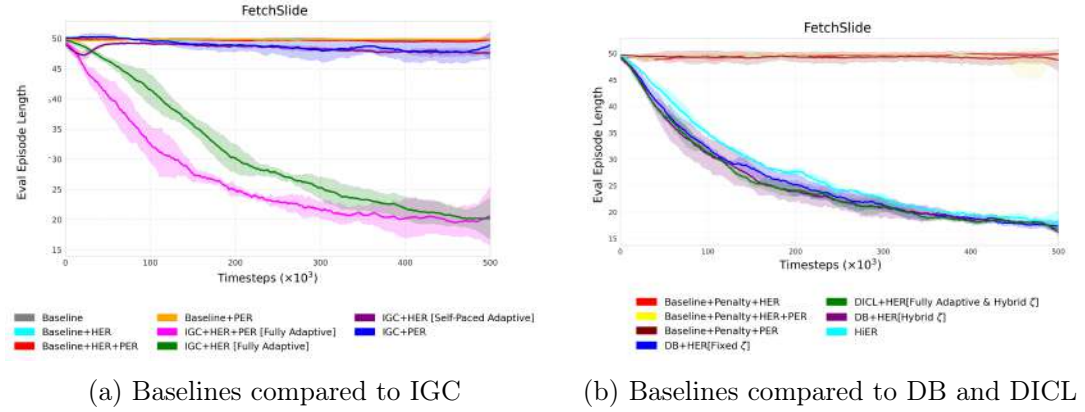


Figure 5.14: Evaluation episode length of Sliding task compared to the Baseline methods for various agent configurations.

5.6 Performance Overview

Among all evaluated configurations, **IGC+HER+PER** demonstrates superior overall performance, with notable robustness in the challenging **Sliding** task. The **DB+HER** variant exhibits the most rapid convergence in both **Pick-and-Place** and **Pushing** environments. Additionally, **DIQL+HER** achieves the highest success rate in **Pick-and-Place** and maintains consistently robust performance across all tasks, underscoring its adaptability and generalization capabilities in diverse robotic settings.

5.7 Qualitative Evaluation

This section presents a qualitative analysis of agent behaviors across the evaluated tasks. Video demonstrations illustrating the performance of our proposed methods are available at the following link¹. These demonstrations correspond to one of the best-performing configurations, which achieves strong overall performance across all tasks. The results highlight the advantages of our approach, including enhanced exploration, faster convergence, and greater reliability in task completion under sparse-reward, continuous control environments.

¹<https://youtu.be/dPOWdrsNT6Y>

Chapter 6

Conclusion

This chapter provides a comprehensive summary of the thesis by revisiting the original motivation, research questions, and the proposed methodologies. It synthesizes the key empirical findings and evaluates the effectiveness of the IGC, the DB mechanism, and their integration within the DICL framework. The discussion emphasizes how these strategies collectively address core challenges in training DRL agents under sparse-reward conditions—specifically, improving sample efficiency, training stability, and convergence speed. The chapter concludes with critical reflections on the strengths and limitations of the proposed approach and outlines potential directions for future research.

6.1 Summary of Thesis Achievements

In this section, we present the principal findings of our study by systematically addressing the research questions introduced in Section 1.3. The answers are substantiated by empirical results and analytical evaluations conducted throughout this research. Each question is revisited below, followed by a concise summary of the corresponding insights derived from our investigation.

RQ1: How can CL strategies be systematically designed and incorporated into model-free DRL algorithms to maximize sample efficiency, convergence speed, and training stability in continuous-control domains?

To systematically design and incorporate CL strategies into model-free DRL algorithms, this thesis proposes novel methods tailored for sparse-reward, continuous-control environments. Specifically, it introduces the Incremental Graded Curriculum

6. Conclusion

(IGC) for structured data collection, the Dual Buffer Mechanism (DB) for curriculum-aware experience replay, and their integration into a unified framework termed Dual-Mechanism Incremental Curriculum Learning (DICL). IGC regulates task complexity by gradually expanding the spatial sampling regions for object and goal placements based on agent performance or training progression. Complementing this, DB implements structured replay by categorizing experiences into positive, negative, and primary buffers, and sampling from them using fixed or adaptive ratios. The DICL framework combines these components into a cohesive architecture that aligns progressive task complexity with structured replay. Together, these strategies address key challenges in sparse-reward, continuous-control environments by improving sample efficiency, accelerating convergence, and enhancing training stability, as validated through extensive experiments on benchmark robotic tasks.

RQ2: What are the comparative advantages and trade-offs between fixed-schedule and adaptive curriculum strategies in terms of convergence speed, asymptotic performance, and robustness across standard benchmark tasks?

The comparative analysis in this research demonstrates that adaptive curriculum strategies generally outperform fixed-schedule approaches across key performance metrics. The adaptive variant of the IGC achieves faster convergence, higher final success rates, and improved training stability across the *FetchPickAndPlace*, *FetchPush*, and *FetchSlide* tasks. By dynamically adjusting task difficulty based on the agent’s success rate, adaptive IGC facilitates a well-aligned and progressive learning trajectory, reducing early-stage failures common in sparse-reward environments. In contrast, the fixed-schedule IGC advances independently of agent performance, which can result in either premature difficulty escalation or prolonged training on overly simple scenarios. In the DB mechanism, fixed sampling ratios yield slightly better outcomes in *FetchPickAndPlace* and *FetchPush* tasks, while the adaptive variant shows marginal gains in the more complex *FetchSlide* task. However, the observed differences across sampling strategies are minor and do not suggest a consistent advantage for either approach. These results suggest that adaptive strategy is particularly beneficial in regulating task complexity during data collection, while both fixed and adaptive sampling can be comparably effective in data exploitation depending on the task.

Overall, there is a trade-off between simplicity and adaptability: fixed methods

6. Conclusion

are easier to implement but less responsive to the agent’s learning progress, while adaptive methods provide greater flexibility by adjusting to learning dynamics in real time, though they require additional monitoring and parameter tuning.

RQ3: How does the combined use of CL and structured experience replay mechanisms influence the exploration–exploitation balance in continuous-control environments?

The DICL framework enables a principled modulation of the exploration–exploitation trade-off through the integration of IGC and DB. IGC promotes structured exploration by gradually increasing environmental complexity, allowing the agent to begin in success-prone subregions before transitioning to more challenging scenarios. This controlled progression helps prevent early policy collapse, particularly in sparse-reward settings. Concurrently, DB enhances exploitation by replaying strategically valuable experiences—both successful outcomes and informative failures—through dedicated positive and negative buffers. Proportional Sampling Ratio strategies further enhance this balance by adjusting replay priorities based on the agent’s learning progress. Together, these mechanisms support stable and progressive policy development, leading to improved performance consistency and generalization across tasks.

RQ4: To what extent do the proposed CL approaches generalize across diverse robotic benchmarks with respect to performance transferability and robustness?

The proposed curriculum learning methods—IGC, DB, and their integration within the DICL framework—exhibit strong generalization across three diverse sparse-reward robotic benchmark tasks: *FetchPickAndPlace*, *FetchPush*, and *FetchSlide*. Notably, all tasks are trained using a single, unified set of hyperparameters, without environment-specific tuning, underscoring the robustness and transferability of the proposed strategies. In relatively simpler tasks such as *FetchPickAndPlace* and *FetchPush*, several configurations achieve 100% success with zero variance across three random seeds. Furthermore, the DB mechanism consistently facilitates faster convergence, highlighting its role in accelerating policy learning. In the most challenging setting, *FetchSlide*, the combination of IGC achieves a 98% success rate, surpassing all baseline methods. These findings confirm that the proposed CL approaches are not only sample-efficient but also broadly applicable across continuous-control tasks with varying levels of difficulty and reward sparsity.

6.2 Future Work

Although the DICL framework has demonstrated marked improvements in sample efficiency and stability within simulated MuJoCo environments, its real-world applicability must be rigorously evaluated. Our first avenue of future work is to deploy DICL on a 7-DoF manipulator operating under realistic conditions. We will leverage extensive domain randomization—varying workspace geometry, lighting conditions, object textures, and sensor noise—during offline training. Subsequently, on-robot fine-tuning will be performed under safety-aware curricula that enforce force and torque limits as well as collision avoidance. Through systematic trials, we will quantify the sim-to-real transfer gap in terms of success rate, convergence speed, and robustness to unmodeled disturbances. This deployment will further allow exploration of curriculum schedules that adapt not only to task difficulty but to hardware wear and environmental drift over time.

Simultaneously, we will extend DICL to collaborative multi-agent scenarios in which two or more manipulators jointly manipulate large or flexible objects. We aim to develop a centralized *curriculum coordinator* that synchronizes difficulty progression across agents and manage shared replay buffers which store both individual successes or failures and joint interaction trajectories. We will investigate how different communication protocols affect cooperative skill acquisition, measuring metrics such as joint task completion time, communication overhead, and resilience to agent failures. Ultimately, this research will demonstrate how adaptive curricula and structured replay can be scaled to multi-robot systems, enabling emergent coordination policies in complex, dynamic environments.

Appendix A

Reinforcement Learning

This appendix presents a concise overview of RL fundamentals relevant to the methods used in this research. It covers key algorithmic categorizations, foundational techniques such as dynamic programming and temporal-difference learning, and the implementation of circular replay buffers. A brief historical perspective on RL in robotics is also included to contextualize the progression toward modern DRL approaches.

A.1 Categorization of RL Algorithms

Model-Based vs. Model-Free

Model-based reinforcement learning methods aim to learn or exploit a model of the environment’s dynamics, specifically the transition and reward function $p(s_{t+1}, r_t \mid s_t, a_t)$. This model is used to simulate future trajectories, allowing the agent to plan actions without requiring real-time interaction. Such methods are advantageous when the dynamics are known or can be efficiently approximated. Notable examples include Dynamic Programming (DP) [17], Dyna-Q [58], and AlphaGo [1]. In contrast, model-free methods bypass the modeling of environment dynamics and instead learn policies or value functions purely through interaction and trial-and-error. This makes them especially suitable for complex or high-dimensional environments where modeling is impractical, such as in robotics. Representative model-free methods include Q-learning [59], SARSA [60], Deep Q-Network (DQN) [3], and a wide range of policy gradient algorithms such as REINFORCE [61], Advantage Actor-Critic (A2C/A3C) [62], Proximal Policy

A. Reinforcement Learning

Optimization (PPO) [33]), Deep Deterministic Policy Gradient (DDPG) [36], Twin Delayed DDPG (TD3) [38], and Soft Actor-Critic (SAC) [39].

On Policy vs Off Policy

On-policy learning (SARSA, PPO) updates the agent’s policy using only the data it actually collects under that policy. This direct approach tends to be stable but requires many interactions to improve. Off-policy learning (Q-Learning, DDPG, TD3, SAC), by contrast, separates the behavior policy (which generates data) from the target policy (which is improved), enabling the agent to reuse past experiences and learn more sample-efficiently. However, off-policy methods must account for the distributional mismatch between the behavior policy (used to collect data) and the target policy (being optimized), which necessitates corrective mechanisms to ensure stable and unbiased updates. Despite this challenge, their ability to reuse past experiences significantly improves sample efficiency. In off-policy actor-critic algorithms, at each environment step, the agent samples an action $a_t \sim \pi_\phi(\cdot | s_t)$, observes a reward r_t , and transitions to the next state s_{t+1} . The resulting transition tuple (s_t, a_t, r_t, s_{t+1}) is then stored in a replay buffer $\mathcal{B}_{\text{ser}}$, typically implemented as a circular buffer (see Appendix A.3), for future updates.

Value-Based vs. Policy-Based vs. Actor-Critic

Value-based methods estimate state-value $v(s)$ or state-action value functions $q(s, a)$ as defined in Equations 2.8 and 2.9, guiding action selection via strategies like ϵ -greedy [13], upper confidence bound (UCB) [63], optimistic initialization [64], or exploring starts [13]. These methods are stable and sample-efficient but generally limited to discrete action spaces. Common algorithms include Q-learning, SARSA, and DQN.

Policy-based methods instead optimize the policy directly, often representing it as a probability distribution over actions. These are well-suited for continuous action spaces but can suffer from high variance and lower sample efficiency. A notable example is REINFORCE [61].

Actor-critic methods integrate both paradigms by using a value-estimating critic to update a policy-improving actor. Though they include value function learning, they are categorized as policy-based due to the explicit policy representation.

Examples include A2C, A3C, PPO, DDPG, TD3, and SAC, widely adopted in continuous control tasks.

A.2 Tabular Reinforcement Learning

Tabular reinforcement learning refers to the class of RL problems in which the state and action spaces are discrete and sufficiently small to allow for the explicit storage of value functions in lookup tables. In this setting, value-based methods are predominantly employed within the framework of *Generalized Policy Iteration (GPI)*, which alternates between *policy evaluation* and *policy improvement* steps until convergence. During policy evaluation, the agent estimates the value function $v^\pi(s)$ or the action-value function $q^\pi(s, a)$ under the current policy π . Subsequently, in the policy improvement step, the policy $\pi(a | s)$ is updated by selecting actions that maximize the estimated value. This iterative enhancement continues until the policy stabilizes and converges to the optimal solution. GPI constitutes the theoretical foundation for classical algorithms such as *policy iteration* and *value iteration*.

A.2.1 Dynamic Programming

Dynamic Programming (DP) [17] provides exact solutions for Markov Decision Processes (MDPs) when the transition dynamics $p(s' | s, a)$ and reward function $r(s, a)$ are fully known. It relies on recursive Bellman updates to compute optimal value functions and policies.

Policy Iteration: This method alternates between evaluating a policy and improving it. The evaluation step estimates the value function for the current policy π_k using the Bellman expectation equation:

$$V^{\pi_k}(s) = \sum_a \pi_k(a | s) \sum_{s'} p(s' | s, a) [r(s, a) + \gamma V^{\pi_k}(s')], \quad (\text{A.1})$$

The improvement step then derives a new policy π_{k+1} by greedily selecting actions with respect to the updated value estimates:

$$\pi_{k+1}(s) = \arg \max_a \sum_{s'} p(s' | s, a) [r(s, a) + \gamma V^{\pi_k}(s')]. \quad (\text{A.2})$$

A. Reinforcement Learning

Value Iteration: In contrast to policy iteration, value iteration merges the evaluation and improvement steps into a single update based on the Bellman optimality equation:

$$V_{k+1}(s) = \max_a \sum_{s'} p(s' | s, a) [r(s, a) + \gamma V_k(s')], \quad (\text{A.3})$$

$$\pi^*(s) = \arg \max_a \sum_{s'} p(s' | s, a) [r(s, a) + \gamma V^*(s')]. \quad (\text{A.4})$$

The algorithm converges to the optimal value function V^* and corresponding optimal policy π^* under finite MDP settings.

A.2.2 Monte Carlo Methods

Monte Carlo (MC) methods [13] estimate value functions by averaging returns from sampled episodes generated under a given policy. These methods are model-free and do not require prior knowledge of the environment's transition dynamics. Unlike DP or TD methods, MC updates rely on complete episodes, with returns computed from actual experiences starting at time step t :

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{T-t-1} R_T. \quad (\text{A.5})$$

MC methods do not employ bootstrapping, making them unbiased but often high in variance. Variants include *first-visit* and *every-visit* MC, depending on whether all or only the first occurrence of a state is used for updates during an episode.

A.2.3 Temporal-Difference Learning

Temporal-Difference (TD) methods [18] blend the advantages of DP and MC by learning directly from experience while using bootstrapping to update estimates incrementally. TD methods are also model-free, requiring no explicit model of the environment. A basic example is TD(0), which updates the value function using the immediate reward and the estimated value of the subsequent state:

$$V(s_t) \leftarrow V(s_t) + \alpha [R_{t+1} + \gamma V(s_{t+1}) - V(s_t)]. \quad (\text{A.6})$$

Unlike MC, TD methods do not require episodes to terminate and allow for online, step-by-step learning. This enables faster reward propagation and improved sam-

ple efficiency, albeit at the cost of introducing bias. Notable TD algorithms include SARSA and Q-learning, and. Techniques such as Double Q-learning mitigate over-estimation bias by decoupling action selection and evaluation across two separate value functions.

A.3 Circular Replay Buffer

The standard replay buffer $\mathcal{B}_{\text{ser}}$ is implemented as a fixed-capacity circular first-in-first-out (FIFO) queue of size C , consistent with prior work [3, 36, 12]. The buffer maintains a write index x and an occupancy counter $n \leq C$. Upon the arrival of a new transition τ_t , the buffer state is updated as follows:

$$\mathcal{B}_{\text{ser}}[x] \leftarrow \tau_t, \quad x \leftarrow (x + 1) \bmod C, \quad n \leftarrow \min(n + 1, C).$$

This design guarantees constant-time insertion and automatically overwrites the oldest transitions once the buffer is full. During training, a mini-batch of size B is sampled by uniformly drawing indices from the interval $\{0, \dots, n - 1\}$, ensuring decorrelated and unbiased transition samples for policy optimization.

A.4 Historical Developments of RL in Robotics

The application of reinforcement learning (RL) to robotics has evolved significantly since the foundational work by Barto, Sutton, and Anderson, who demonstrated that trial-and-error controllers could solve classic tasks such as pole balancing and cart-pole stabilization [65, 13]. In the early 1990s, Moore and Atkeson introduced temporal-difference (TD) learning to address inverse kinematics in a 6-DOF robotic arm [66], while Atkeson et al. conducted comparative analyses of model-based and model-free approaches for sparse-reward trajectory learning [67]. Kaelbling, Littman, and Moore surveyed a wide range of RL applications in robotics—including navigation, grasping, and control—emphasizing challenges such as low sample efficiency and limited exploration in high-dimensional continuous spaces [68].

Around the same time, Tesauro’s TD-Gammon illustrated the potential of neural function approximators for sequential decision-making [69], inspiring policy gradient methods such as those later introduced by Peters and Schaal [70]. Follow-up reviews by Kober et al. further underscored the sim-to-real transfer problem [71]

A. Reinforcement Learning

and advocated for hierarchical [72] and adaptive-critic [73] frameworks to enhance generalization. Notably, PILCO [74] advanced probabilistic model-based RL with high sample efficiency, while learning-from-demonstration approaches [75, 76] significantly accelerated real-world policy learning.

In recent years, multi-agent reinforcement learning (MARL) has become a critical subfield, enabling coordinated behaviors in multi-robot systems. Canese et al. [77] surveyed key MARL challenges—such as non-stationarity, credit assignment, and scalability—and classified existing methods into centralized training with decentralized execution (CTDE), value decomposition, and communication-based models. Their work also illustrated real-world applications, including robot soccer, drone swarms, and cooperative manipulation. Offline RL, as exemplified by the D4RL benchmark [78], has emerged to enable policy learning from static datasets without real-time interaction—an important development for safety-critical domains.

Parallel innovations include language-conditioned RL, where agents interpret natural language instructions to perform complex manipulation tasks [79]; self-supervised exploration strategies based on model disagreement to improve sample efficiency [80]; and unified evaluation suites like Meta-World [81] that benchmark generalization across diverse robotic skills. Collectively, these contributions form the theoretical and empirical foundation for modern DRL frameworks that integrate expressive neural models with efficient exploration, structured replay, and robust sim-to-real transfer mechanisms.

Appendix B

Soft Actor–Critic (SAC)

Soft Actor–Critic (SAC) [39] represents a significant advancement in off-policy actor–critic RL, particularly within the domain of continuous control. As with other actor–critic frameworks, SAC comprises a policy network (actor) and a value function estimator (critic), which are optimized jointly to improve both decision quality and sample efficiency. However, unlike conventional actor–critic algorithms that typically optimize the expected cumulative reward, SAC is derived from the maximum entropy RL formulation, which introduces an entropy regularization term into the objective. This addition explicitly encourages policy stochasticity and enables more robust exploration strategies. The learning objective in SAC is defined as:

$$J(\pi) = \mathbb{E}_{\pi} \left[\sum_{t=0}^{\infty} \gamma^t (R_{t+1} + \alpha \mathcal{H}(\pi(\cdot \mid s_t))) \right], \quad (\text{B.1})$$

where $\mathcal{H}(\pi(\cdot \mid s)) = -\mathbb{E}_{a \sim \pi} [\log \pi(a \mid s)]$ denotes the Shannon entropy of the policy, and α is a temperature parameter that modulates the importance of the entropy term. By maximizing both expected return and entropy, SAC yields policies that retain high action diversity, thereby mitigating the risk of early convergence to suboptimal deterministic behaviors—particularly prevalent in environments with sparse-reward structures. The critic is trained using a soft variant of the Bellman equation. Given a transition tuple (s_i, a_i, r_i, s'_i) , the soft Q-target is defined as:

$$y_i = r_i + \gamma \mathbb{E}_{a' \sim \pi_{\phi}} [Q_{\bar{\theta}}(s'_i, a') - \alpha \log \pi_{\phi}(a' \mid s'_i)], \quad (\text{B.2})$$

where $Q_{\bar{\theta}}$ denotes the target Q-function, which is updated using Polyak averaging to ensure stability. The policy parameters are updated by minimizing a stochastic

B. Soft Actor–Critic (SAC)

approximation to the entropy-regularized expected Q-value:

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{s \sim \mathcal{D}, a \sim \pi_\phi} [\alpha \log \pi_\phi(a \mid s) - Q_\theta(s, a)]. \quad (\text{B.3})$$

To stabilize learning and reduce overestimation bias, SAC employs a twin-Q architecture and a soft update rule for the target networks:

$$\bar{\theta} \leftarrow \tau \theta + (1 - \tau) \bar{\theta}, \quad (\text{B.4})$$

where $\tau \in (0, 1)$ controls the update rate. These components, when integrated within the entropy-maximizing framework, enable SAC to achieve state-of-the-art performance on a broad range of continuous control benchmarks.

Appendix C

Prioritized Experience Replay (PER)

Prioritized Experience Replay (PER) improves sample efficiency by prioritizing the replay of transitions with higher temporal-difference (TD) errors (see Fig. 3.1 for the workflow). Unlike uniform sampling, PER biases the selection process toward more informative transitions, as indicated by the magnitude of their TD errors.

Each transition i is assigned a scalar priority:

$$p_i = |\delta_i| + \varepsilon, \quad (\text{C.1})$$

where δ_i is the TD error, and $\varepsilon > 0$ is a small constant ensuring all transitions maintain a non-zero probability of being sampled. The TD error is defined as:

$$\delta_i = r_i + \gamma Q_{\theta'}(s_{i+1}, \pi(s_{i+1})) - Q_{\theta}(s_i, a_i), \quad (\text{C.2})$$

where $Q_{\theta'}$ is the target critic network, and $\pi(s_{i+1})$ is the next action selected by the current policy.

Transitions are sampled from the standard replay buffer $\mathcal{B}_{\text{ser}}$ according to the probability distribution:

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}, \quad (\text{C.3})$$

where $\alpha \in [0, 1]$ controls the degree of prioritization. When $\alpha = 0$, the sampling becomes uniform; as α increases, sampling becomes more skewed toward higher-priority transitions.

To correct for the bias introduced by prioritized sampling, importance-sampling (IS) weights are used:

$$w_i = \frac{(NP(i))^{-\beta}}{\max_j (NP(j))^{-\beta}}, \quad (\text{C.4})$$

C. Prioritized Experience Replay (PER)

where N is the buffer size, and $\beta \in [0, 1]$ is annealed from an initial value β_0 to 1 over the course of training. These weights reduce the variance of updates and help maintain unbiased expectations.

During training, a mini-batch of transitions $\{(s_i, a_i, r_i, s'_i, d_i)\}$ is sampled based on $P(i)$. The critic minimizes the following weighted TD loss:

$$y_i = r_i + \gamma(1 - d_i) Q_{\theta'}(s'_i, \pi(s'_i)), \quad \mathcal{L}_Q = \sum_i w_i (Q_\theta(s_i, a_i) - y_i)^2. \quad (\text{C.5})$$

Following the gradient update, new TD errors $|\delta_i|$ are computed for the sampled transitions, and the corresponding priorities p_i in the buffer are updated. This dynamic adjustment allows PER to continuously focus on the most informative experiences, accelerating learning and improving convergence stability.

Appendix D

Hindsight Experience Replay (HER)

Hindsight Experience Replay (HER) [11] is a replay-augmentation technique designed to improve sample efficiency in sparse-reward reinforcement learning environments. Rather than discarding trajectories in which the agent fails to achieve the original goal, HER transforms these failures into learning opportunities by re-labeling them with alternative goals that were actually achieved during the episode (see Fig. 3.2).

At the end of each episode, HER processes the collected transitions $(s_t, a_t, r_t, s_{t+1}, d_t)$ to generate additional synthetic transitions. This is done by selecting an alternative goal g' —typically chosen from a future state in the same episode trajectory—and then modifying the transition by replacing the original goal g with g' . The reward r'_t and terminal condition d'_t are recomputed based on the new goal, resulting in a relabeled transition $(s_t, a_t, r'_t, s_{t+1}, d'_t)$.

These relabeled transitions are stored in the same replay buffer $\mathcal{B}_{\text{ser}}$ alongside the original ones. During training, transitions are sampled uniformly from this augmented buffer, enabling the agent to learn from both real and hindsight experiences. This decouples the relabeling process from the sampling procedure and avoids computational overhead during policy updates.

Several goal-selection strategies have been proposed for choosing g' , including: *Future*: select a goal from a future state in the same episode, *Final*: use the final state of the episode as the goal, *Next*: use the next state as the goal, *Episode-Random*: sample a random state within the episode. Among these, the *Future* strategy has shown superior empirical performance due to its balance between diversity and goal feasibility.

Bibliography

- [1] David Silver et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. In: *arXiv preprint arXiv:1712.01815* (2017).
- [2] David Silver et al. “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. DOI: 10.1038/nature24270.
- [3] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533. DOI: 10.1038/nature14236.
- [4] Matthew Hausknecht and Peter Stone. “Deep recurrent Q-learning for partially observable MDPs”. In: *arXiv preprint arXiv:1507.06527* (2015).
- [5] Christopher Berner et al. “Dota 2 with large scale deep reinforcement learning”. In: *arXiv preprint arXiv:1912.06680* (2019).
- [6] Oriol Vinyals et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: *nature* 575.7782 (2019), pp. 350–354.
- [7] Yan Duan et al. “Benchmarking deep reinforcement learning for continuous control”. In: *International conference on machine learning*. PMLR. 2016, pp. 1329–1338.
- [8] Yoshua Bengio et al. “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48.
- [9] Sanmit Narvekar et al. “Curriculum learning for reinforcement learning domains: A framework and survey”. In: *Journal of Machine Learning Research* 21.181 (2020), pp. 1–50.
- [10] Petru Soviany et al. “Curriculum learning: A survey”. In: *International Journal of Computer Vision* 130.6 (2022), pp. 1526–1565.

BIBLIOGRAPHY

- [11] Marcin Andrychowicz et al. “Hindsight experience replay”. In: *Advances in neural information processing systems* 30 (2017).
- [12] Tom Schaul et al. “Prioritized experience replay”. In: *arXiv preprint arXiv:1511.05952* (2015).
- [13] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. Vol. 1. 1. MIT press Cambridge, 1998.
- [14] Carlos Florensa et al. “Automatic goal generation for reinforcement learning agents”. In: *arXiv preprint arXiv:1705.06366* (2017).
- [15] Tom Schaul et al. “Universal value function approximators”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1312–1320. URL: <https://proceedings.mlr.press/v37/schaul15.html>.
- [16] Ashvin V Nair et al. “Visual reinforcement learning with imagined goals”. In: *Advances in neural information processing systems* 31 (2018).
- [17] Richard Bellman. “Dynamic programming”. In: *science* 153.3731 (1966), pp. 34–37.
- [18] Richard S Sutton. “Learning to predict by the methods of temporal differences”. In: *Machine learning* 3 (1988), pp. 9–44.
- [19] Xin Wang, Yudong Chen, and Wenwu Zhu. “A survey on curriculum learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.9 (2021), pp. 4555–4576.
- [20] Carlos Florensa, Yan Duan, and Pieter Abbeel. “Reverse curriculum generation for reinforcement learning”. In: *Proceedings of the 1st Conference on Robot Learning (CoRL)*. PMLR. 2017, pp. 482–495.
- [21] Vitchyr H Pong et al. “Skew-fit: State-covering self-supervised reinforcement learning”. In: *arXiv preprint arXiv:1903.03698* (2019).
- [22] David Held et al. “Automatic goal generation for reinforcement learning agents”. In: (2018).

BIBLIOGRAPHY

- [23] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. “Automatic curriculum learning through value disagreement”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7648–7659.
- [24] Andrew Y Ng, Daishi Harada, and Stuart J Russell. “Policy invariance under reward transformations: Theory and application to reward shaping”. In: *Proceedings of the 16th International Conference on Machine Learning (ICML)*. Morgan Kaufmann. 1999, pp. 278–287.
- [25] Adrien Baranes and Pierre-Yves Oudeyer. “Active learning of inverse models with intrinsically motivated goal exploration in robots”. In: *Robotics and Autonomous Systems* 61.1 (2013), pp. 49–73.
- [26] Rémy Portelas et al. “Automatic curriculum learning for deep rl: A short survey”. In: *arXiv preprint arXiv:2003.04664* (2020).
- [27] Tamber Matsen et al. “Teacher-Student Curriculum Learning”. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. AAAI. 2019, pp. 4936–4943.
- [28] Alex Graves et al. “Automated curriculum learning for neural networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 1311–1320.
- [29] Carlos Florensa et al. “Automatic goal generation for reinforcement learning agents”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. PMLR. 2018, pp. 1515–1528.
- [30] Rémy Portelas et al. “Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 835–853.
- [31] Andrej Karpathy. *Curriculum learning and starting small*. <http://karpathy.github.io/2016/05/31/rl/>. Accessed: 2024-03-23. 2016.
- [32] Guy Hach Cohen and Daphna Weinshall. “On the power of curriculum learning in training deep networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 2535–2544.
- [33] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

BIBLIOGRAPHY

- [34] Riashat Islam et al. “Reproducibility of benchmarked deep reinforcement learning tasks for continuous control”. In: *arXiv preprint arXiv:1708.04133* (2017).
- [35] OpenAI: Marcin Andrychowicz et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.
- [36] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).
- [37] Peter Henderson et al. “Deep reinforcement learning that matters”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [38] Scott Fujimoto, Herke Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: *International conference on machine learning*. PMLR. 2018, pp. 1587–1596.
- [39] Tuomas Haarnoja et al. “Soft actor-critic algorithms and applications”. In: *arXiv preprint arXiv:1812.05905* (2018).
- [40] Lerrel Pinto et al. “Asymmetric actor critic for image-based robot learning”. In: *arXiv preprint arXiv:1710.06542* (2017).
- [41] Ivan Koryakovskiy et al. “Benchmarking model-free and model-based optimal control”. In: *Robotics and Autonomous Systems* 92 (2017), pp. 81–90.
- [42] Kazumi Kasaura et al. “Benchmarking actor-critic deep reinforcement learning algorithms for robotics control with action constraints”. In: *IEEE Robotics and Automation Letters* 8.8 (2023), pp. 4449–4456. DOI: 10.1109/LRA.2023.3280815.
- [43] Denis Yarats et al. “Improving sample efficiency in model-free reinforcement learning from images”. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 10674–10681.
- [44] Boris Ivanovic et al. “Barc: Backward reachability curriculum for robotic reinforcement learning”. In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 15–21.
- [45] Tim Salimans and Richard Chen. “Learning montezuma’s revenge from a single demonstration”. In: *arXiv preprint arXiv:1812.03381* (2018).

BIBLIOGRAPHY

- [46] Sebastien Racaniere et al. “Automated curricula through setter-solver interactions”. In: *arXiv preprint arXiv:1909.12892* (2019).
- [47] Rémy Portelas et al. “Meta automatic curriculum learning”. In: *arXiv preprint arXiv:2011.08463* (2020).
- [48] Cédric Colas et al. “Curious: intrinsically motivated modular multi-goal reinforcement learning”. In: *International conference on machine learning*. PMLR. 2019, pp. 1331–1340.
- [49] Dániel Horváth et al. “HiER: Highlight Experience Replay for Boosting Off-Policy Reinforcement Learning Agents”. In: *IEEE Access* (2024).
- [50] Junhyuk Oh et al. “Self-imitation learning”. In: *International conference on machine learning*. PMLR. 2018, pp. 3878–3887.
- [51] Meng Fang et al. “Curriculum-guided hindsight experience replay”. In: *Advances in neural information processing systems* 32 (2019).
- [52] Jesus Bujalance and Fabien Moutarde. “Reward Relabelling for combined Reinforcement and Imitation Learning on sparse-reward tasks”. In: *22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*. 2023.
- [53] Johan Ferret, Olivier Pietquin, and Matthieu Geist. “Self-imitation advantage learning”. In: *arXiv preprint arXiv:2012.11989* (2020).
- [54] Che Wang and Keith Ross. “Boosting soft actor-critic: Emphasizing recent experience without forgetting the past”. In: *arXiv preprint arXiv:1906.04009* (2019).
- [55] Emanuel Todorov, Tom Erez, and Yuval Tassa. “MuJoCo: A physics engine for model-based control”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033. DOI: 10.1109/IRoS.2012.6386109.
- [56] OpenAI. *Soft Actor-Critic — Spinning Up Documentation*. <https://spinningup.openai.com/en/latest/algorithms/sac.html>. Accessed: Jul. 13, 2023. 2023.
- [57] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. Chapman and Hall/CRC, 1994.

BIBLIOGRAPHY

- [58] Richard S Sutton. “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming”. In: *Machine learning proceedings 1990*. Elsevier, 1990, pp. 216–224.
- [59] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8 (1992), pp. 279–292.
- [60] Gavin A Rummery and Mahesan Niranjana. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.
- [61] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8 (1992), pp. 229–256.
- [62] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. PmLR. 2016, pp. 1928–1937.
- [63] Peter Auer. “Using confidence bounds for exploitation-exploration trade-offs”. In: *Journal of Machine Learning Research* 3.Nov (2002), pp. 397–422.
- [64] Richard S Sutton. “Generalization in reinforcement learning: Successful examples using sparse coarse coding”. In: *Advances in neural information processing systems* 8 (1995).
- [65] Andrew G Barto, Richard S Sutton, and Charles W Anderson. “Neuronlike adaptive elements that can solve difficult learning control problems”. In: *IEEE transactions on systems, man, and cybernetics* 5 (1983), pp. 834–846.
- [66] Andrew W Moore and Christopher G Atkeson. “Prioritized sweeping: Reinforcement learning with less data and less time”. In: *Machine learning* 13 (1993), pp. 103–130.
- [67] Andrew Atkeson and Patrick J Kehoe. “Modeling and measuring organization capital”. In: *Journal of political Economy* 113.5 (2005), pp. 1026–1053.
- [68] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. “Reinforcement learning: A survey”. In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.
- [69] Gerald Tesauro. “Td-gammon: A self-teaching backgammon program”. In: *Applications of neural networks*. Springer, 1995, pp. 267–285.

BIBLIOGRAPHY

- [70] Jan Peters and Stefan Schaal. “Reinforcement learning of motor skills with policy gradients”. In: *Neural networks* 21.4 (2008), pp. 682–697.
- [71] Andrei A Rusu et al. “Progressive neural networks”. In: *arXiv preprint arXiv:1606.04671* (2016).
- [72] Andrew G Barto and Sridhar Mahadevan. “Recent advances in hierarchical reinforcement learning”. In: *Discrete event dynamic systems* 13 (2003), pp. 341–379.
- [73] Kumpati S Narendra and Kannan Parthasarathy. “Identification and control of dynamical systems using neural networks”. In: *IEEE Transactions on neural networks* 1.1 (1990), pp. 4–27.
- [74] Marc Deisenroth and Carl E Rasmussen. “PILCO: A model-based and data-efficient approach to policy search”. In: *Proceedings of the 28th International Conference on machine learning (ICML-11)*. 2011, pp. 465–472.
- [75] Brenna D Argall et al. “A survey of robot learning from demonstration”. In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.
- [76] Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. “Reinforcement learning in robotics: Applications and real-world challenges”. In: *Robotics* 2.3 (2013), pp. 122–148.
- [77] Lorenzo Canese et al. “Multi-agent reinforcement learning: A review of challenges and applications”. In: *Applied Sciences* 11.11 (2021), p. 4948.
- [78] Justin Fu et al. “D4rl: Datasets for deep data-driven reinforcement learning”. In: *arXiv preprint arXiv:2004.07219* (2020).
- [79] Simon Stepputtis et al. “Language-conditioned imitation learning for robot manipulation tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 13139–13150.
- [80] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. “Self-supervised exploration via disagreement”. In: *International conference on machine learning*. PMLR. 2019, pp. 5062–5071.
- [81] Tianhe Yu et al. “Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning”. In: *Conference on robot learning*. PMLR. 2020, pp. 1094–1100.

14

Supervised Learning for Robot Manipulation
Tanakrit Lertcompeesin

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING

MASTER'S THESIS No. 141

Supervised Learning for Robot Manipulation

Tanakrit Lertcompeesin

Zagreb, June 2025.

I would like to sincerely thank the IFROS program, the University of Girona, and the University of Zagreb for offering me the valuable opportunity to pursue this master's program. I am grateful for the extensive knowledge and support provided by their distinguished faculty members throughout my studies.

My deepest gratitude goes to my supervisor, Prof. Narcís Palomeras, for his expert guidance, constant support, and thoughtful feedback throughout the course of this thesis.

I would also like to extend special thanks to Hayat Hussein for his valuable technical expertise and assistance, as well as to Prof. Tamara Petrović for her representation and support from the University of Zagreb.

Finally, I am thankful to all my colleagues and friends whose encouragement and emotional support have been a source of strength during this journey.

CONTENTS

1. Introduction	1
1.1. Scope	2
1.2. Contributions	2
1.3. Outline	3
2. Literature Review and Background Theory	4
2.1. Literature Review	4
2.1.1. Robot Learning in Manipulation	4
2.1.2. Imitation Learning	5
2.1.3. Behavior Cloning	6
2.1.4. DAGGER Algorithm	7
2.1.5. Supervised Learning Architecture	7
2.1.6. Sim2Real Transfer	8
2.2. Theoretical Framework	8
2.2.1. Behavior Cloning as Supervised Learning	8
2.2.2. ResNet	9
2.2.3. ResNet3D	11
2.2.4. DAGGER Algorithm	12
3. Methodology	14
3.1. Expert Demonstration	15
3.1.1. Task Definition	15
3.1.2. Oracle Controller	16
3.1.3. Domain Randomization	17
3.2. Data Collection	20
3.2.1. Demonstration Dataset Description	20
3.2.2. Data Collection Pipeline	21
3.3. Behavior Cloning via Supervised Learning	22

3.3.1. Model Architecture	22
3.3.2. Training Procedure	25
3.4. Policy Refinement with DAgger	27
3.4.1. Modifications to Expert Behavior	28
3.4.2. DAGGER Process	29
4. Simulation Framework and Real Robot Deployment	31
4.1. Simulation Environment	31
4.1.1. MuJoCo Simulation	31
4.2. Simulation Scene Setup	32
4.3. Real Robot Deployment	34
4.3.1. Interface with UR3e	34
4.3.2. Real Robot Setup	36
4.3.3. DAGGER on the Real Robot	37
5. Results	39
5.1. Behavior Cloning Experiment	40
5.2. DAGGER Experiment	43
5.3. Real Robot Experiment	47
6. Conclusion	51
6.1. Future Work	52
Bibliography	54

1. Introduction

One of the core challenges in robotics is enabling robots to perform tasks in real-world environments with human-like adaptability and precision. Traditional control systems are typically based on pre-programmed rules or simple control strategies, which work well for straightforward tasks like pick-and-place operations. However, these approaches often depend on highly accurate object localization, something that is rarely available in unstructured, real-world settings.

A common method for perceiving the environment is through visual input, which provides rich information, resembles human perception, and is relatively affordable. Despite these advantages, it is difficult to directly use visual data for decision-making in complex tasks. Translating raw images into precise actions is not straightforward, especially for fine manipulation. In addition, visual input suffers from variability and noise in real-world scenarios, often causing traditional computer vision and image processing pipelines, which attempt to extract object positions, to fail.

Consider a situation where a reliable control policy already exists in a simulated environment, operating effectively using precise, simulation-provided information. The remaining challenge is to relate raw visual input to the control actions. One promising approach to address this is imitation learning, a branch of robot learning that focuses on learning from expert demonstrations.

This paper explores how various techniques of imitation learning can be used to teach robots to understand and act on visual input by leveraging simulation. The core idea is to treat the expert controller in simulation as a teacher. While the expert performs the task using perfect state information, we record the corresponding visual observations. Simulation allows us to collect large amounts of data efficiently and to apply domain randomization to create a variety of visual scenarios. This helps the learned model generalize better and become more robust when exposed to the diversity of real-world images.

The collected data is then used to train a control policy through supervised

learning, an approach known as behavior cloning in the context of imitation learning [1]. The full pipeline involves setting up a simulation environment for a simple assembly task, developing an expert controller to execute the task, collecting training data that includes both visual inputs and corresponding actions, training an initial policy using behavior cloning, and further improving the model using the DAGGER algorithm [2]. DAGGER mitigates the issue of compounding errors during deployment by allowing the model to learn from its own mistakes in an iterative fashion. Once the model has satisfactory performance in simulation, it can then be transferred to the real robot for execution.

By following this imitation learning pipeline, the robot can learn to perform tasks based on visual input alone, building a foundation for deploying more adaptable and visually capable robots in the real world.

1.1. Scope

This research investigates the use of imitation learning to train a robotic manipulator to perform an assembly task using visual input and proprioceptive data such as end-effector position and joint states. The UR3e robot is used as the manipulation platform throughout the project. The control policy is first trained in simulation using behavior cloning and then refined with the DAGGER algorithm. Domain randomization is applied to improve visual robustness and generalization. The final model is deployed on the UR3e robot in a real-world setting to evaluate the effectiveness of sim-to-real transfer. The study focuses on RGB vision and proprioception, excluding other sensing modalities such as force or tactile feedback.

1.2. Contributions

1. Proposed an imitation learning pipeline that avoids human demonstrations by using an expert system with privileged access to scene information, which is only available in simulation.
2. Modification and evaluation of supervised model architectures for visuomotor control, tailored to the requirements of the assembly task and informed by the limitations observed in baseline models.

3. Deployment and real-world evaluation of the trained model to assess sim-to-real transfer performance and validate the effectiveness of domain randomization and imitation learning.

1.3. Outline

This thesis is organized as follows. Chapter 2 reviews the existing literature on robot learning, focusing on imitation learning, behavior cloning, DAGGER, supervised learning, and domain randomization in manipulation tasks. It also presents the theoretical framework used throughout the research. Chapter 3 describes the methodology and techniques applied in this study. Chapter 4 details the implementation process, covering both the simulation environment and the deployment on the real UR3e robot. Chapter 5 presents the experimental results, including comparisons of different model architectures, improvements achieved through DAGGER, and performance evaluations of the final model in both simulation and real-world settings. Finally, Chapter 6 concludes the thesis and discusses possible future work.

2. Literature Review and Background Theory

2.1. Literature Review

2.1.1. Robot Learning in Manipulation

Robotic manipulation tasks, particularly those involving contact-rich interactions, high degrees of freedom, and vision-based perception, remain highly challenging due to the complexity of the dynamics, the uncertainty in visual input, and the precision required for successful execution. Traditional control approaches struggle to generalize beyond narrowly defined conditions, making them brittle in the face of real-world variability. Recent advances in learning-based methods have shown promise in addressing these challenges by enabling robots to acquire manipulation skills directly from data. [3]

Especially through the application of Reinforcement Learning (RL), robot learning has made significant progress in manipulation tasks. In RL, agents learn to interact with their environment by maximizing cumulative rewards based on feedback from their actions. This approach is well-suited for tasks involving high degree of freedom and trial-and-error learning. For example, OpenAI's work on dexterous in-hand manipulation used Proximal Policy Optimization (PPO) to train a five-fingered robotic hand in simulation, which was then successfully transferred to the real world using domain randomization techniques [4]

However, RL is not a perfect solution, particularly when applied to real-world robotic systems. One significant challenge is the **long training times** required for RL to converge to a stable and efficient policy, as the agent must repeatedly interact with the environment, often taking thousands or even millions of steps. Furthermore, RL faces the **problem of safe learning**, especially in environments where mistakes can lead to dangerous consequences, such as in robotic manipulation

tasks or autonomous driving [3].

2.1.2. Imitation Learning

Given the limitations of RL in terms of training time and safety, **Imitation Learning (IL)** has emerged as an alternative approach. Imitation learning allows robots to learn from expert demonstrations, leveraging **supervised learning techniques** to replicate the behavior of an expert. This approach significantly reduces the need for extensive exploration and trial-and-error learning, which are inherent challenges in RL.

To combine the strengths of both approaches, some methods use demonstrations to bootstrap the learning process and then refine the policy through reinforcement learning. One such approach is Q-learning from Demonstrations (DQfD), where the agent is first pretrained using supervised learning on expert data and then continues to learn by interacting with the environment, gradually improving its policy while still leveraging the original demonstrations [5].

On the other hand, some imitation learning approaches like Behavior Cloning (BC) aim to learn the task entirely from demonstration data, without any interaction with the environment. BC treats imitation as a supervised learning problem, using a single deep neural network to map observations to actions. Despite its simplicity, several works have successfully used BC to acquire manipulation skills, especially in structured tasks where demonstrations are sufficient to capture the variability in the environment [6].

With the advancement of machine learning techniques, Generative Adversarial Networks (GANs) inspired new directions in imitation learning. One notable method is Generative Adversarial Imitation Learning (GAIL), which formulates imitation as a game between a generator (the policy) and a discriminator. The policy aims to produce actions indistinguishable from those of the expert, while the discriminator learns to tell them apart. GAIL has shown greater sample efficiency than BC and better generalization in complex environments [7]. This framework has also been applied to robotic manipulation, for instance, in the context of high-precision tasks such as the assembly of pegs in the hole from human demonstrations [8].

2.1.3. Behavior Cloning

Behavior Cloning (BC) is one of the simplest and most widely used techniques in imitation learning. In BC, a model, usually a deep neural network, is trained using supervised learning to directly map observations, such as images or proprioceptive data, to the corresponding actions taken by an expert. This method treats imitation as a supervised learning problem, with the goal of minimizing the difference between the model’s predicted actions and those demonstrated by the expert. Behavior cloning has its roots in early work such as Pomerleau’s ALVINN system [1], which used a neural network to learn autonomous vehicle steering by mimicking human behavior based on visual input.

The effectiveness of behavior cloning has been demonstrated in real-world manipulation tasks, particularly when combined with simulation and domain randomization. In the work by Levine et al. [9], a pick-and-place behavior was successfully learned and deployed on a real robot using only behavior cloning from data generated in simulation. The study highlighted that, with sufficient training data and carefully designed randomization, the model achieved strong visual robustness, enabling effective sim-to-real transfer. This idea is further supported by James et al. [10], who showed that end-to-end visuomotor policies trained entirely in simulation could perform complex manipulation tasks such as rope shaping and stacking in the real world. By improving the model architecture and leveraging large-scale simulated data, the robot was able to acquire diverse skills while requiring minimal real-world fine-tuning.

Behavior cloning performs exceptionally well when trained on expert or near-optimal demonstrations, enabling fast learning without the need for exploration. However, a key limitation is its vulnerability to *distributional shift*, where the robot encounters states during execution that were not represented in the training data. This often leads to compounding errors and degraded policy performance, especially in dynamic or uncertain environments [11, 12]. Additionally, in many practical scenarios, demonstrations are collected from humans, which can be time-consuming and difficult to scale. The limited availability of high-quality demonstrations often results in insufficient training data, making behavior cloning less applicable in real-world settings compared to other learning methods that rely on exploration or reinforcement learning.

2.1.4. DAGGER Algorithm

As mentioned earlier, policies learned through behavior cloning can make errors during deployment, especially when they encounter unfamiliar states. These mistakes can lead the policy to drift further from the expert trajectory, compounding errors over time. This issue limits the generalization and robustness of policies trained solely with standard behavior cloning. To address this, the DAGGER algorithm was introduced in [2]. DAGGER iteratively improves the policy by allowing it to interact with the environment and collect data from its own trajectory. In each iteration, the current policy generates new states, and the expert provides the correct actions for those states. This aggregated dataset is then used to retrain the policy, reducing the distribution mismatch between training and deployment.

Despite its effectiveness, the original DAGGER algorithm assumes the continuous availability of an expert to label every state visited by the policy, which can be impractical and resource-intensive. To address this limitation, several extensions of DAGGER have been proposed. One such variant, SafeDagger [13], reduces the labeling burden by selectively querying the expert only when the policy's confidence in its decision is low. Another extension, AggreVaTe [14], incorporates cost-to-go information into the learning process, allowing the policy to focus more effectively on long-term outcomes and improving learning efficiency in complex tasks.

2.1.5. Supervised Learning Architecture

CNNs have been widely applied in robotic learning due to their ability to extract spatial features from raw images. For example, [9] demonstrated end-to-end training in visuomotor policies using CNNs for manipulation tasks such as opening and grasping doors. Similarly, [15] applied CNN-based architectures in a meta-learning framework to enable one-shot visual imitation learning, allowing a robot to replicate manipulation tasks from a single video demonstration. More recently, Vision Transformers (ViTs) have gained attention for their ability to model long-range dependencies in visual data. [16] introduced a transformer-based model that performs visual foresight by predicting future frames to guide manipulation. In addition, [17] used CLIP-based Vision Transformers to encode semantic visual information, enabling manipulation with minimal supervision. These works illustrate the effectiveness of CNNs and ViTs in visual processing for imitation learning

in robotic manipulation tasks. Another important architecture is a Recurrent Neural Network (RNN), especially variants such as LSTMs which are vital for capturing temporal dependencies in robotic manipulation tasks. They help the model remember past states, which is crucial for multi-stage tasks where current decisions depend on previous context. For instance, [9] showed that removing LSTMs caused failures in long-horizon tasks. By modeling task progression over time, RNNs improve the robustness of visuomotor policies when visual information alone is ambiguous.

2.1.6. Sim2Real Transfer

One of the core challenges in robot learning is the discrepancy between simulated environments and the real world—commonly known as the sim-to-real gap. This gap is particularly evident in manipulation tasks, where mismatches in dynamics, object properties, and especially visual input can significantly degrade performance. Simulations often provide clean, idealized images, whereas real-world environments are affected by noise, lighting variations, reflections, and background clutter. As a result, policies trained purely in simulation may fail to generalize when deployed on real robots [18].

A popular approach to mitigate this gap is domain randomization, which involves training the model in simulations where visual and physical properties are randomized. By varying aspects like textures, colors, lighting, camera angles, and object positions, the model learns to focus on robust features that generalize across domains. This technique has shown promising results in enabling policies to transfer from simulation to reality without fine-tuning [19, 18]. It is particularly effective in visual imitation learning, where generating large, diverse training datasets in simulation can compensate for the limited availability of real-world demonstrations.

2.2. Theoretical Framework

2.2.1. Behavior Cloning as Supervised Learning

Behavior Cloning (BC) is a subset of *imitation learning* that frames the problem of policy learning as a **supervised regression** or **classification** task. Given a dataset of state-action pairs generated by an expert demonstrator, the goal is to train a

parametric policy π_θ that approximates the expert's behavior by minimizing a predefined loss function.

Let $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ be a dataset where:

- $s_i \in \mathcal{S}$ denotes the **state** (e.g., sensor observations, pixel inputs),
- $a_i \in \mathcal{A}$ denotes the **action** (e.g., continuous control, discrete decisions) taken by the expert in state s_i .

The policy $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$ is trained to minimize the empirical risk:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(a_i, \pi_\theta(s_i)),$$

where $\mathcal{L}$ is a task-specific loss function:

- For **continuous actions** (e.g., steering angles), $\mathcal{L}$ is typically the *mean squared error* (MSE):

$$\mathcal{L}(a, \pi_\theta(s)) = \|a - \pi_\theta(s)\|_2^2.$$

- For **discrete actions** (e.g., categorical choices), $\mathcal{L}$ is the *cross-entropy loss*:

$$\mathcal{L}(a, \pi_\theta(s)) = - \sum_{j=1}^K a_j \log \pi_\theta(s)_j,$$

where K is the number of discrete actions.

The success of BC relies on two key assumptions:

1. **Expert Optimality:** The demonstrations in $\mathcal{D}$ must be near-optimal; otherwise, the cloned policy will inherit suboptimal biases.
2. **IID State Distribution:** The training states s_i must be *independently and identically distributed* (IID) with respect to the states encountered during policy execution. Violations (e.g., due to *covariate shift*) lead to compounding errors, as analyzed in [1].

2.2.2. ResNet

ResNet (Residual Network) is a convolutional neural network architecture designed to address the degradation problem in deep networks by introducing identity-based residual connections. Originally proposed in [20], ResNet allows models to be significantly deeper without the risk of vanishing gradients by using shortcut connections that skip one or more layers. The basic building block of ResNet is a

residual block, which computes the output as a sum of the input and a learned residual function. ResNet also comes pretrained on large-scale datasets such as ImageNet, which provides a strong initialization that helps improve feature extraction and speeds up convergence during training. For these reasons, ResNet is chosen as the vision backbone in this thesis. This choice is further supported by [10], which successfully employs ResNet for a similar visuomotor learning task.

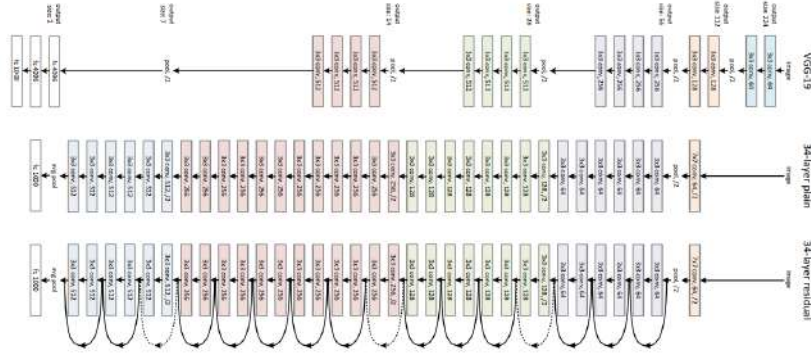


Figure 2.1: ResNet-34 architecture. Source: [20].

Table 2.1: Comparison of ResNet Variants

ResNet Variant	Depth (Layers)	Approx. Parameters (Millions)
ResNet-18	18	11.7
ResNet-34	34	21.8
ResNet-50	50	25.6
ResNet-101	101	44.5
ResNet-152	152	60.2

The naming convention of ResNet architectures (e.g., ResNet-18, ResNet-50) directly reflects their depth in layers and corresponding parameter counts as shown in 2.1. The parameter difference primarily stems from two block designs:

- BasicBlock (used in ResNet-18/34): Two 3×3 convolutional layers
- Bottleneck (used in ResNet-50+): Three convolutional layers ($1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$) that dramatically reduce parameters through dimensionality reduction

2.2.3. ResNet3D

ResNet-3D extends traditional residual learning to spatiotemporal data by replacing 2D convolutions with 3D convolutions. These 3D filters operate across height, width, and time dimensions, making the architecture well-suited for tasks involving dynamic visual input, such as object tracking or understanding motion in manipulation tasks.

Two ResNet-3D variants, proposed in [21], have been investigated to balance performance and efficiency. The first variant, called the Mixed Convolution (MC) architecture, uses 3D convolutions in the lower layers to model motion and switches to 2D convolutions in higher layers to handle spatial reasoning. This design reflects the idea that motion cues are best captured early in the network, while also reducing the parameter overhead incurred by 3D convolutions. Consequently, MC ResNets achieve performance close to full 3D ResNets while using significantly fewer parameters.

The second variant, known as (2+1)D convolutions, factorizes a 3D convolution into a sequence of a 2D spatial convolution followed by a 1D temporal convolution. This decomposition takes inspiration from 2D depthwise separable convolutions [21] to reduce the parameter overhead of full 3D convolutions. While doing so, it introduces an extra non-linearity between the operations, enabling the model to also learn more expressive features. It also simplifies optimization, typically resulting in lower training and testing loss.

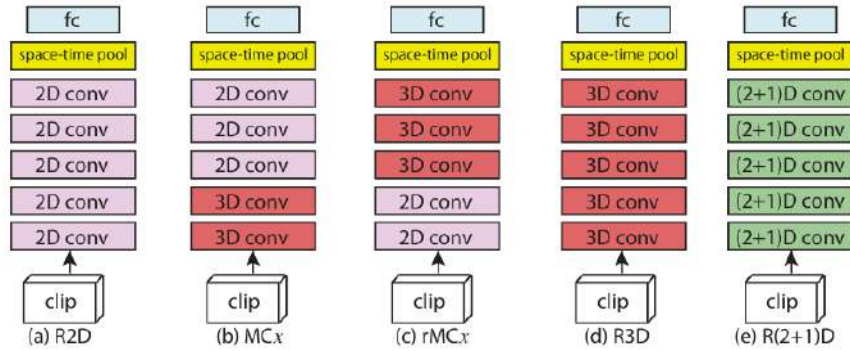


Figure 2.2: (a) R2D are 2D ResNets; (b) MCx are ResNets with mixed convolutions (MC3 is presented in this figure); (c) rMCx use reversed mixed convolutions (rMC3 is shown here); (d) R3D are 3D ResNets; and (e) R(2+1)D are ResNets with (2+1)D convolutions. Source: [21]

Figure 2.2 illustrates the architectural differences between the standard 3D

ResNet, Mixed Convolution, and (2+1)D variants.

2.2.4. DAGGER Algorithm

Dataset Aggregation (DAGGER) is a well-known imitation learning algorithm designed to address the limitations of standard behavior cloning, particularly the issue of distributional shift. In behavior cloning, a policy is trained on expert demonstrations, which leads to a mismatch between the training data distribution (states visited by the expert) and the states encountered at test time (visited by the learned policy). This mismatch often causes compounding errors during deployment. DAGGER mitigates this by allowing the learned policy to interact with the environment, while continuously querying the expert to correct its mistakes.

Formally, let π^* be the expert policy and π_θ the learned policy with parameters θ . The goal is to minimize the expected loss over the state distribution induced by π_θ :

$$\mathbb{E}_{s \sim d_{\pi_\theta}} [\mathcal{L}(\pi_\theta(s), \pi^*(s))]$$

where d_{π_θ} is the distribution of states encountered by following policy π_θ , and $\mathcal{L}$ is a supervised loss function, such as mean squared error between predicted and expert actions. In contrast, standard behavior cloning minimizes the loss over d_{π^*} , which does not reflect the learner's actual experience.

DAGGER addresses this by iteratively collecting data under the learner's policy while labeling each observation with the expert's action and updating the training dataset. At each iteration, the learner gathers states by acting in the environment and queries the expert to label those states. These expert-labeled examples are added to the dataset, and the policy is retrained. Over time, the dataset better reflects the states encountered during deployment, thus reducing distributional shift.

Algorithm 1 DAGGER (Dataset Aggregation)

Initialize dataset $\mathcal{D}_1 \leftarrow \emptyset$

Initialize policy $\pi_1 \leftarrow \pi^*$ (the expert)

$i = 1$ to N Run policy π_i to collect trajectory τ_i

state s in τ_i Query expert for $a^* = \pi^*(s)$

Add (s, a^*) to dataset: $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_i \cup \{(s, a^*)\}$

Train new policy π_{i+1} on dataset $\mathcal{D}_{i+1}$ using supervised learning

This iterative data aggregation approach enables the learner to progressively reduce its performance gap with the expert. DAGGER has been shown to enjoy no-regret guarantees, meaning the average regret of the sequence of learned policies decreases over time. However, a practical limitation is the need to query the expert at every iteration, which may be expensive or infeasible in real-world settings. Several extensions such as SafeDagger and AggreVaTe have been proposed to address this, reducing expert queries by introducing confidence thresholds or using cost-to-go feedback to guide learning [13, 14].

3. Methodology

In this chapter, we present the methodology used to develop an imitation learning pipeline that enables a robot to acquire visuomotor skills through demonstration. An overview of the entire process is illustrated in figure 3.1. The pipeline begins with the creation of expert demonstrations that solve a basic assembly task using full access to the ground-truth object positions. During the execution of these tasks, we collect both proprioceptive states of the robot and RGB images from three different viewpoints, which together form the input to our learning model.

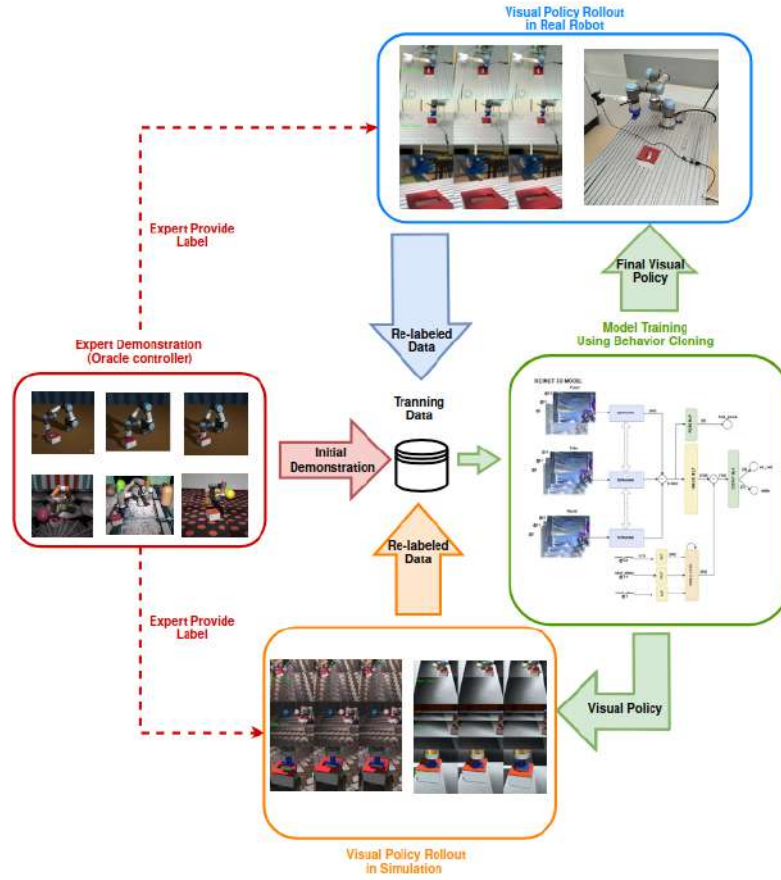


Figure 3.1: Imitation learning pipeline

Next, we explore and train a range of supervised learning architectures to determine which performs best in mapping observations to actions. To address the limitations of behavior cloning, particularly its inability to generalize well to unseen states, we adopt the DAGGER algorithm. This involves deploying the initial policy, collecting new trajectories as the robot encounters novel situations, querying the expert for correct actions in those states, and aggregating the original dataset with these new examples. The updated dataset is then used to retrain the policy iteratively.

Finally, we deploy the trained model in both simulation and on a real UR3e robot to evaluate its performance and assess the effectiveness of the sim-to-real transfer. The rest of this chapter follows these steps in sequence, detailing the implementation and design choices made at each stage.

3.1. Expert Demonstration

In this section, we describe the development of the expert demonstrator, which is responsible for generating the initial training dataset and providing corrective actions during later training stages.

3.1.1. Task Definition

This research focuses on evaluating the feasibility of an imitation learning pipeline rather than solving highly precise or advanced manipulation tasks. Therefore, we selected a relatively simple assembly task that emphasizes sequential execution over fine-grained precision or what we refer to as a "long-horizon task." Specifically, the robot must insert a rectangular peg, attached to its end-effector, into a matching hole and then rotate it 90 degrees once fully inserted. This task serves as a foundational step toward more complex operations, such as turning a key to open a door. The setup of the task in Mujoco simulation is shown in figure 2. The details of the scene layout and task components will be discussed in the following chapter.

In order to achieve this simple assembly task, the target behavior is divided into three sequential states 3.2:

1. **Move to Hole:** The robot must move the peg from its initial position to a point 10 cm above the hole, while also aligning the orientation to prepare for insertion.

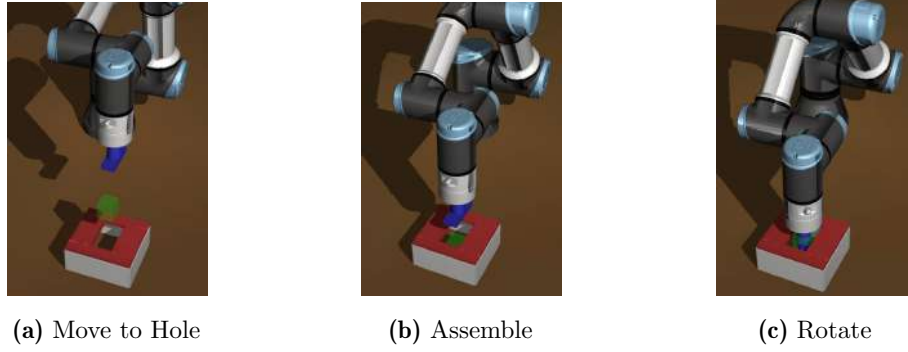


Figure 3.2: The three stages of the assembly task: (a) approaching the hole, (b) inserting the peg, and (c) rotating inside the hole. Green boxes indicate the target positions.

2. **Assemble:** In this phase, the robot lowers the peg into the hole. If any misalignment occurs during descent (common in learned models but not with the expert controller), the robot must adjust its pose before continuing. We will explore this challenge in detail in a later section.
3. **Rotate:** After successful insertion, the final step is to rotate the peg 90 degrees within the hole.

3.1.2. Oracle Controller

The term *oracle controller* refers to an expert policy that has access to privileged information about the environment and can reliably complete the task. In our case, the oracle controller requires the ground-truth pose of the hole, which is available in simulation, to successfully execute the task.

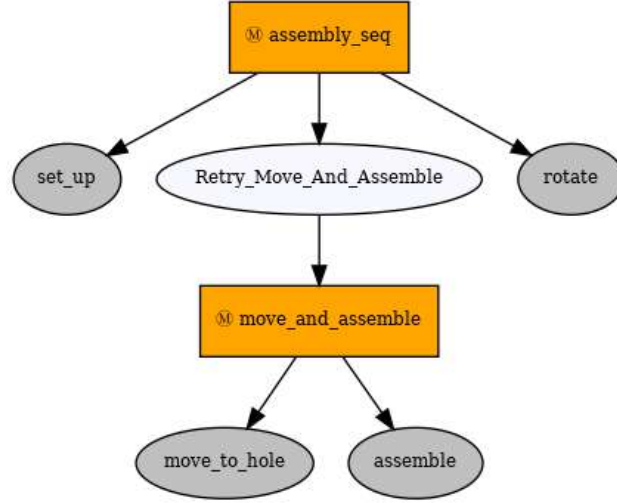


Figure 3.3: Behavior tree for assembly process

We implemented the oracle controller using a behavior tree, as illustrated in figure 3.3. The controller takes the current end-effector pose and the hole pose as inputs. Based on these, it determines the current task state and computes the corresponding target position. The behavior tree handles transitions across different phases of the task, which are defined in the previous subsection. The computed target position is then converted into an end-effector velocity command using a simple feedforward control strategy. This command is sent to the robot through its built-in velocity controller.

During the initial demonstration phase, the actions produced by the oracle controller are used exclusively to control the robot and serve as ground-truth labels for training data, including both velocity commands and task state annotations. In later stages, particularly during the DAGGER training phase, the oracle controller is used to provide corrective actions when the learned policy visits unfamiliar or suboptimal states, as shown in Figure 3.9.

3.1.3. Domain Randomization

Domain randomization is a key technique used to bridge the gap between simulation and real-world deployment. Since our expert demonstrations are repeatable and succeed consistently regardless of visual variations, and because simulations allow easy modification of scene parameters, we leverage domain randomization to increase visual diversity during training. This encourages the model to become visually robust when deployed in the real world, as supported by the results in

[19].

Our randomization strategy is inspired by the approaches in [9, 10], which share a similar methodology. In each demonstration, we randomize the following scene components:

- **Hole Position and Orientation:** The hole’s position is randomly sampled within the workspace boundaries while ensuring no initial collision with the peg. Its orientation is randomized along the yaw axis within the range $[-\pi/2, \pi/2]$, introducing a diverse set of assembly scenarios for the model to learn from.
- **Peg End-Effector Position and Orientation:** Similar to the hole, the peg’s position is randomized within the workspace, ensuring it does not intersect with the hole. Orientation is randomized in yaw $[-\pi, \pi]$, and in pitch and roll $[-\pi/4, \pi/4]$. The robot arm moves to the sampled pose before each demonstration.
- **Arm Base Height:** The base height is uniformly sampled from a small range, with the minimum being just above the table surface. This helps mitigate height discrepancies between simulation and real-world robot setups.
- **Peg and Hole Color:** Colors are sampled from a normal distribution centered around real-world estimates. We perturb the hue, saturation, and brightness (HSV values in $[0, 1]$) within a bounded range $[-\phi_o, \phi_o]$, helping the model generalize under different lighting conditions and surface materials.
- **Camera Parameters:** Camera position, orientation, and field of view (FOV) are randomized around default settings. The camera always looks at the workspace center, which is also slightly randomized. This allows tolerance for imprecise camera placement in real deployment.
- **Light Source:** A point light source is positioned randomly on a portion of a sphere surrounding the workspace. Distances range from 1.0 to 3.0 meters, with azimuthal angles in $[0, \pi/2]$ and polar angles in $[\pi/10, 4\pi/10]$. Ambient lighting is adjusted with noise sampled from $[-\psi_l, \psi_l]$ added to a base ambient coefficient of 0.3. This introduces a variety of realistic lighting and shadow conditions.
- **Scene Textures (Table, Arm, Wall, Floor):** Background textures are randomly sampled from the Describable Textures Dataset (DTD) [22],

which contains 5640 images across 47 perceptual texture categories. All non-objective surfaces are randomized, helping the model ignore irrelevant background details.

- **Distractor Objects:** Random primitive shapes with varying colors, sizes, and positions are added outside the workspace so they do not interfere with task execution. This encourages the model to focus on relevant objects and ignore background clutter.

An example of the applied domain randomization setup used during training is shown in Figure 3.4. This includes randomized textures, lighting, object positions, and distractor elements to ensure the model is exposed to a wide range of perceptual conditions during training.

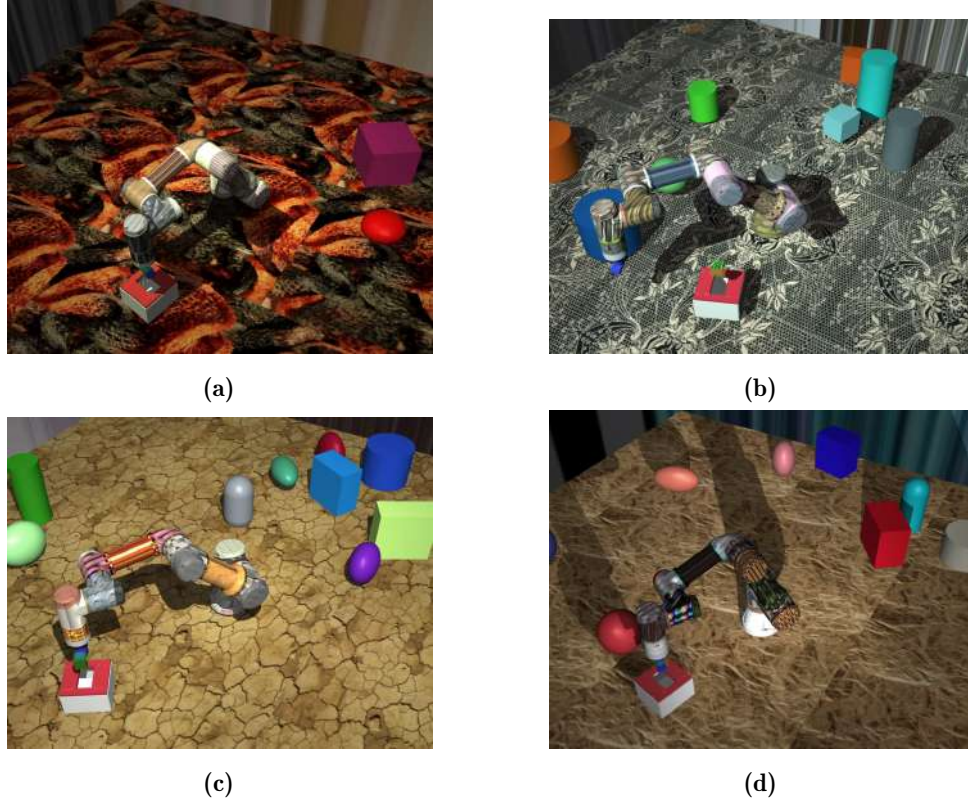


Figure 3.4: Examples of domain randomization applied during simulation training. Each image shows variations in lighting, textures, object positions, and distractors.

3.2. Data Collection

This section describes the structure of our training dataset, the procedure for collecting the data, and the preprocessing steps necessary to prepare it for model training.

3.2.1. Demonstration Dataset Description

The dataset consists of temporally aligned proprioceptive and visual inputs, along with corresponding target outputs. The inputs are defined as follows:

- **Image#1 ($3 \times 3 \times 244 \times 244$):** RGB images from the front fixed camera, captured at three time steps ($t - 2$, $t - 1$, t).
- **Image#2 ($3 \times 3 \times 244 \times 244$):** RGB images from the side fixed camera at the same three time steps.
- **Image#3 ($3 \times 3 \times 244 \times 244$):** RGB images from the in-hand camera at $t - 2$, $t - 1$, and t .
- **End-Effector Pose (3×7):** The Cartesian position and orientation (in quaternion format) of the end-effector with respect to the robot base, for three time steps.
- **Joint States (3×6):** Joint angles of the robot arm at three time steps.

The corresponding outputs are:

- **End-Effector Velocity (1×6):** Linear and angular velocity of the end-effector with respect to the world frame $[v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]$.
- **Hole Pose (1×7):** The pose of the hole in the scene, represented by position and quaternion orientation.
- **Assembly State (1×1):** A discrete label indicating the current stage of the task: move (0), assemble (1), or rotate (2).

Motivation for Input Design: The proprioceptive data (joint states and end-effector pose) give the model awareness of the robot’s internal state. The visual inputs from multiple views (front, side, and in-hand cameras) are critical for understanding the 3D spatial relationships between the peg and hole. The in-hand camera, in particular, provides close-up views for fine alignment, especially when other cameras have obstructed views. Figure 3.5 shows an example of these visual inputs.

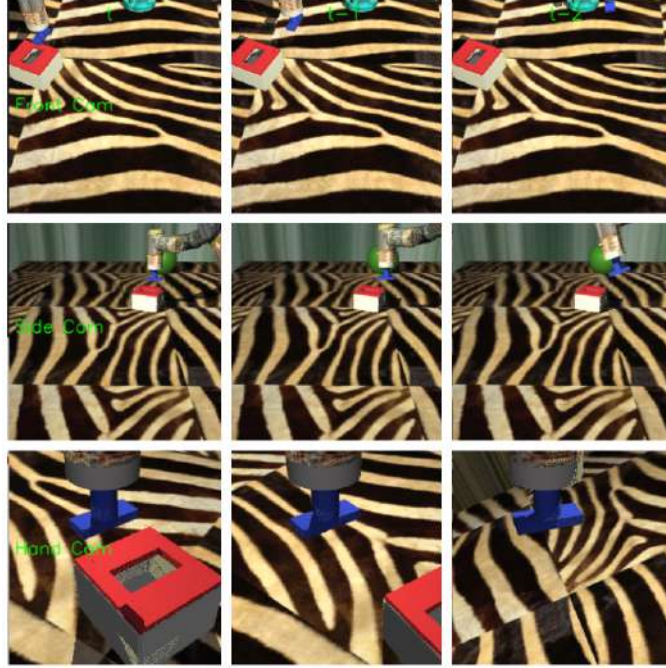


Figure 3.5: Example of visual input data used in the model. It includes stacked RGB frames from three viewpoints: front camera, side camera, and in-hand camera, each containing three consecutive frames at times $t - 2$, $t - 1$, and t .

Use of Temporal Information: Including frames from three consecutive time steps helps the model understand temporal progression. This is important for disambiguating task phases that may appear visually similar at a single point in time.

Output Rationale: The primary model output is the end-effector velocity, which serves as the robot’s action. The hole pose and assembly state predictions are used only during training as auxiliary targets to aid in learning, as supported in [9]. These outputs help ensure that the model truly understands the task context.

3.2.2. Data Collection Pipeline

Data is collected from the simulation at regular intervals. Mujoco operates with a time resolution of 0.002 seconds per step, which would produce redundant data if logged continuously. Therefore, we record one data point every 100 simulation steps, resulting in a sampling rate of 0.2 seconds per data point. Each demonstration sequence contains between 45 and 60 data points, corresponding to 9 to 12 seconds of simulation time per demonstration, depending on scenario

complexity.

Each sample is saved as a `.pkl` file and compressed into a single `.zip` archive per demonstration. The initial dataset consists of approximately 4,000 demonstrations, collected over about 20 hours of simulation time, including around 5 seconds of randomization per demonstration. Since our server runs slightly faster than real-time, the actual collection time is closer to 14 hours.

To prepare the data for training, we implemented a custom dataset class compatible with PyTorch’s `DataLoader`. This class handles loading from `.zip` files, stacking inputs across multiple time steps, and formatting the data appropriately. A `frame_skipped` parameter controls the temporal gap between frames in each sample (currently set to 5, equivalent to a 1-second gap between input frames). This parameter is subject to further optimization based on model performance.

3.3. Behavior Cloning via Supervised Learning

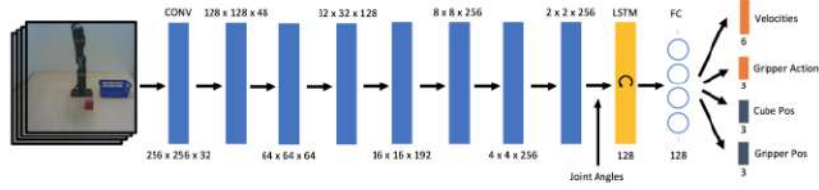
This section discusses the structure of our proposed model, architectural variations we explored, and training details.

3.3.1. Model Architecture

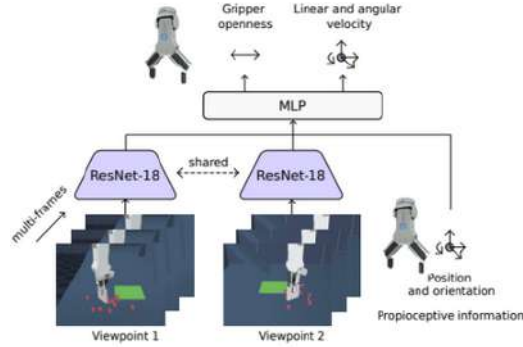
The architecture used in this work is inspired by the designs presented in [9] and [10]. As shown in Figures 3.6a and 3.6b, both models follow a similar structure: a visual module processes input images, and its outputs are concatenated with the robot’s proprioceptive state before being passed through fully connected layers to generate predictions.

Building upon this foundation, we introduce several modifications tailored to the specifics of our task. These changes are summarized below:

- **Vision Module:** The goal of this component is to extract meaningful features from visual inputs taken from multiple viewpoints. Initially, we employed a 2D ResNet-18 architecture, as suggested in [10]. However, we found that this setup underperformed for our task due to the insufficiency in learning strong temporal correlations between successive frames. To address this, we explored the use of 3D ResNet architectures, which have demonstrated their effectiveness in capturing temporal information from sequential image frames [21].



(a) Model architecture from [9]

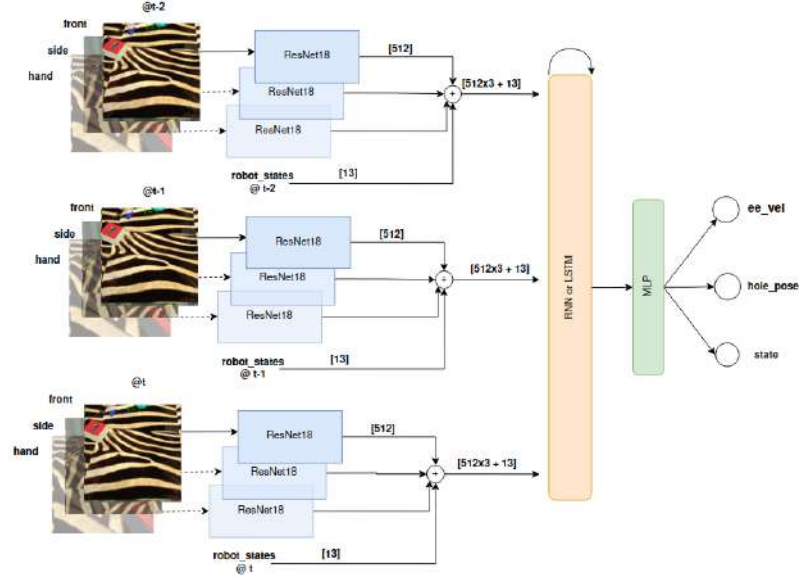


(b) Model architecture from [10]

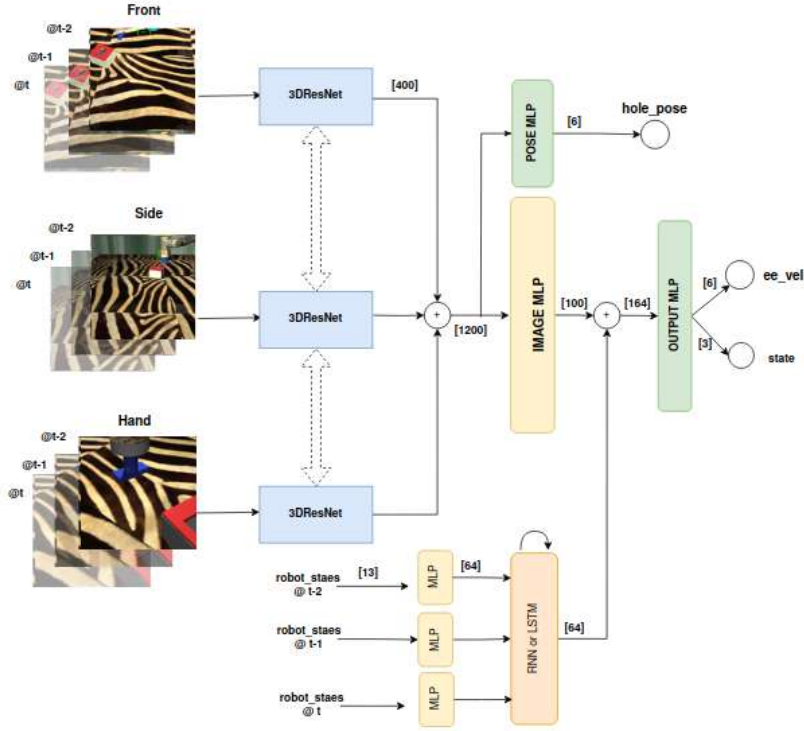
Figure 3.6: Model architectures from prior works that inspired our approach.

- **Hole Position as Intermediate Output:** In contrast to [9], where the hole position is predicted at the final output layer, we propose predicting it as an intermediate output. This change is motivated by the observation that predicting hole position using the end-effector state can confuse the model, since the hole location should primarily depend on visual inputs. Making it an intermediate target encourages the visual encoder to extract more spatially relevant information.
- **Temporal Modeling:** As the assembly task involves sequential stages, modeling temporal dependencies is crucial not only for the visual representations but also for the robot state sequences. Our proposed architecture incorporates an LSTM to capture temporal cues from the robot’s proprioceptive input, while the 3D ResNet variants process image streams.

Figure 3.7a illustrates our initial architecture, referred to as the *Vanilla Model*. This model uses ResNet18 to extract visual features from each input image. The feature representations from different viewpoints are then concatenated along with the robot state vector. The combined representation is passed through an LSTM module to capture temporal dependencies across the input sequence. Finally, the LSTM output is fed into a multi-layer perceptron (MLP) to produce the action predictions.



(a) Vanilla Model Architecture



(b) Late Fusion Model Architecture

Figure 3.7: Comparison of the two architectures developed in this work. The Vanilla Model serves as the baseline, while the Late Fusion Model represents our final architecture, incorporating the modifications mentioned.

Building on this, we introduced aforementioned enhancements that led to our final architecture, referred to as the *Late Fusion Model*, which independently processes the spatio-temporal features from the multi-view image sequences and the temporal cues from the state sequences before fusing the learned representations. This is as shown in figure 3.7b.

This final model uses a multi-channel 3D ResNet as the visual encoder for each viewpoint. The 3D convolution captures temporal dynamics in the sequence of input frames. The resulting three visual embeddings (one per camera) are then used to predict the hole pose as an intermediate output. These embeddings are then fused into a single representation.

For the robot’s proprioceptive input (end-effector pose and joint states), we first project it into a higher-dimensional space using an MLP to better match the scale of the visual features and prevent it from being overshadowed. An RNN is then applied to capture temporal dependencies in the robot state sequence. The resulting robot-state representation is concatenated with the fused image representation, and the combined vector is passed through another MLP to predict the final outputs: the desired end-effector velocity and predicted assembly’s state.

3.3.2. Training Procedure

The primary objective of our model is to predict the end-effector velocity, which serves as the robot’s action output. In addition, we include two auxiliary outputs—hole pose estimation and assembly state classification—not for inference use, but to assist the training process and serve as debugging signals. These outputs help the model develop a better internal understanding of the task, as suggested in prior work [9]. Accordingly, we define separate loss functions for each output: Mean Squared Error (MSE) is used for both the velocity and hole pose (regression tasks), and Cross-Entropy Loss is used for the state classification (a discrete task). The total training loss is computed as a weighted sum:

$$\mathcal{L}_{\text{total}} = \lambda_v \mathcal{L}_{\text{vel}} + \lambda_p \mathcal{L}_{\text{pose}} + \lambda_s \mathcal{L}_{\text{state}}$$

where $\lambda_v, \lambda_p, \lambda_s$ are scalar weights. Since velocity prediction is our main objective, we assign a significantly higher weight to λ_v , while λ_p and λ_s are set to smaller values to ensure they support but do not dominate the training process.

We train the model using the dataset collected from simulation, as detailed in Section 3.2. The dataset is randomly split, with 80% allocated for training

Algorithm 2 Training Loop for UR3E Behavioral Cloning

```

1: Input: Dataset  $\mathcal{D}$ , Model  $M$ , Epochs  $N$ , Learning rate  $\alpha$ , Batch size  $B$ 
2: Initialize optimizer and learning rate scheduler
3: if Resuming training then
4:   Load model and optimizer from checkpoint
5: end if
6: Split  $\mathcal{D}$  into training and validation sets
7: for epoch = 1 to  $N$  do
8:   Set  $M$  to training mode
9:   for each batch  $(x,y)$  in training set do
10:    Compute predictions  $\hat{y} = M(x)$ 
11:    Compute weighted loss:  $L = \lambda_v L_{\text{vel}} + \lambda_p L_{\text{pose}} + \lambda_s L_{\text{state}}$ 
12:    Backpropagate and update model
13:   end for
14:   Save model checkpoint
15:   Set  $M$  to evaluation mode
16:   for each batch  $(x,y)$  in validation set do
17:    Compute predictions  $\hat{y} = M(x)$ 
18:    Evaluate loss and classification accuracy
19:   end for
20: end for

```

and 20% for validation. The training process employs fixed hyperparameters, summarized in Table 3.1, which were chosen based on common practice rather than through extensive tuning. Future work may explore systematic hyperparameter optimization to enhance performance. The training procedure follows a standard supervised learning approach, as outlined in Algorithm 2.

Table 3.1: Training Hyperparameters

Parameter	Value	Notes
Optimizer	Adam	–
Initial Learning Rate	1×10^{-4}	With polynomial decay (PolyLR)
Batch Size	20	–
Training Epochs	20	No early stopping
Frame Skip	5	$\approx 1s$ interval between frames
Loss Weights	$[0.87, 0.087, 0.043]$	$[\lambda_v, \lambda_p, \lambda_s]$

During training, we monitor both training and validation losses for each individual output to ensure balanced learning and detect potential overfitting. While loss metrics provide useful signals about the optimization process, they do not fully capture the model’s actual behavior in the task environment. In our case, a low loss does not necessarily guarantee successful task execution. Therefore, in addition to tracking loss curves, we periodically perform rollouts using the trained model to qualitatively evaluate its performance. These rollouts offer a more practical measure of the model’s ability to generalize and behave correctly in real scenarios.

3.4. Policy Refinement with DAgger

This section describes the limitations observed in the initial policy trained via behavioral cloning, the modifications made to the expert behavior to address these issues, and the procedure we followed to refine the policy using the DAgger algorithm. We explain how the new data is collected, aggregated, and used for further training.

3.4.1. Modifications to Expert Behavior

We observed that the trained model often fails during the assembly phase, particularly when inserting the peg into the hole. The most common failure occurs due to slight misalignments inherited from the previous step (move phase), which causes the peg to miss the hole entirely. Instead of readjusting, the model continues to push downward, resulting in failed insertions. These misaligned states are outside the distribution of the training data and were not visited during expert demonstrations, meaning the model lacks exposure to them. Figure 3.8 illustrates some of these failure scenarios.



Figure 3.8: Examples of failure scenarios during the assembly phase where the trained model fails to insert the peg due to misalignment.

To address this issue, we modified the expert policy during the insertion phase. Instead of directly moving down, the expert first checks whether the current position and orientation are within a threshold range that allows successful insertion. If not, it adjusts its alignment—either through planar movements or rotational corrections—before proceeding downward.

Another issue emerged from our original assumption that once the robot reaches the insertion phase, it would not drift significantly from the hole. However, in practice, the learned policy sometimes drifts away after transitioning to the insertion phase. In such cases, the original expert continues giving insertion actions even when the robot is too far from the hole—actions that no longer make sense. To resolve this, we introduced a fallback mechanism: if the robot drifts beyond a predefined threshold, the expert deems the insertion state as failed and switches back to the move state, guiding the robot to re-approach the hole.

These additions do not compromise the integrity of our training pipeline. The newly introduced expert behaviors only apply to failure cases that never occurred during the original demonstrations. Hence, they can be seen as enriching the dataset with new scenarios rather than contradicting the existing data.

3.4.2. DAGGER Process

To refine the policy, we follow the standard DAGGER procedure (Algorithm 1). First, we deploy the learned policy in the same simulated environment used for collecting the initial demonstrations, with domain randomization applied. While the model is performing the task, we record the input at each timestep, paired with labels provided by the modified expert policy as described in Section 3.2. The key difference is that the robot acts according to the learned policy, not the expert. Figure 3.9 illustrates the data recollection process. We collect approximately 400-500 new demonstrations per DAGGER iteration.

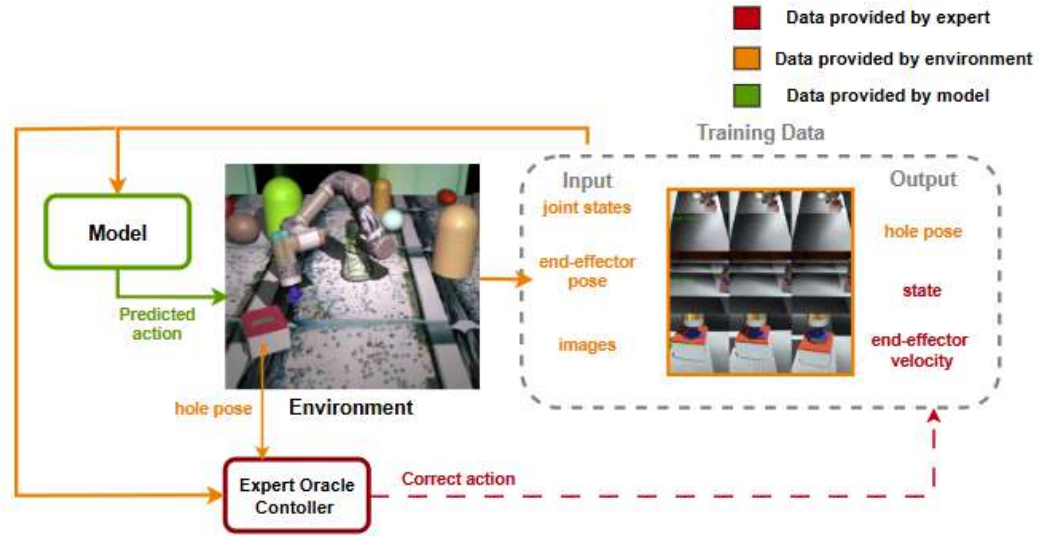


Figure 3.9: Illustration of the data recollecting process during DAGGER. The trained model is deployed in the environment, and the expert policy provides the corrected labels for each visited state.

Once new data is collected, we aggregate it with the original training dataset. Retaining the original data is important to prevent the model from overfitting to the new distribution. While techniques such as data weighting or selective sampling can be applied to balance old and new data, we found that directly aggregating all data yielded satisfactory results. Therefore, no additional sampling strategies were used in our work.

After aggregation, we retrain the model using the updated dataset. Instead of training from scratch, we continue training from the weights of the model used in the current DAGGER iteration. The training hyperparameters are the same as those in Table 3.1, except for a reduced number of epochs (14 instead of 20) to

prevent overfitting.

Each DAGGER iteration consists of three steps: data recollection (collecting 400-500 new demonstrations), data aggregation (combining these with the initial pool of 4,000 samples), and retraining the whole dataset (for 14 epochs). After each iteration, we evaluate the updated model to measure performance improvements. However, since each iteration takes about 26-30 hours to complete—4 hours for data collection and 22-26 hours for retraining—we limited the DAGGER refinement to the "Late Fusion Model," which demonstrated the best initial performance. The final model underwent 5 iterations of DAGGER, resulting in a total dataset of approximately 6,500 demonstrations. The entire process—including data recollection, aggregation, and retraining—spanned roughly 5 days.

4. Simulation Framework and Real Robot Deployment

This chapter presents the development environment used for both simulation and real-world deployment. It includes an overview of the MuJoCo simulator used during training and testing, as well as the hardware and software configuration of the real UR3e robot system.

4.1. Simulation Environment

This section introduces the MuJoCo physics engine and describes how the simulated environment was constructed for training purposes.

4.1.1. MuJoCo Simulation

In this thesis, the task is first simulated using MuJoCo simulator. Developed by Google DeepMind, MuJoCo (stands for Multi-Joint dynamics with Contact) [23] is a high-performance physics simulator designed for simulating articulated structures with complex contacts, such as robotic manipulators and biomechanical models. It is highly optimized for speed and accuracy in simulating complex, contact-rich interactions, making it well-suited for robotic control tasks. Its integration with common DRL libraries, such as OpenAI Gym and Stable-Baselines3, further supports rapid prototyping and benchmarking. Additionally, MuJoCo supports customization of environments and is widely used in academic benchmarks, contributing to its popularity in the learning-based robotics community.

To create a simulated environment in MuJoCo, this thesis builds upon the template provided by the open-source *Manipulator-Mujoco* repository [24]. This repository offers a Gymnasium-based environment structure that can be adapted for reinforcement learning tasks. It is built on top of the *dm_control* library

[25], which is a Python-based interface and environment suite developed by Google DeepMind for research in continuous control and reinforcement learning. The *Manipulator-Mujoco* repository also includes helpful tools such as a generic operational space controller and transformation utilities, which were used to support the development of the custom assembly environment implemented in this thesis.

The implemented custom environment follows the format of a Gymnasium environment class [26], which contains the following components:

- **Initialization**

The environment is initialized by setting up the task, the manipulator arm, the observation space, and the action space, along with some other variables and tools related to the learning algorithm, such as the domain randomizer and learning curriculum. This function also contains rendering options in order to visualize the environment in MuJoCo simulator.

- **Reset**

This method is called at the beginning of every episode to reset the environment. This includes setting the manipulator back to its initial configuration and randomizing the position and orientation of the hole. Once the environment is reset, this method returns the initial observation.

- **Step**

This method is called to perform a single time step iteration inside the environment. A step consists of performing an action, computing the state of the environment after applying that action, computing the reward at that step, and evaluating termination criteria. This method returns a 5-tuple consists of observation, reward, terminated, truncated, and info.

- **Render**

This method contains the functionalities to visualize the states of the environment with the MuJoCo simulator. It is essential for visualizing the training process and the performance of the trained policy.

4.2. Simulation Scene Setup

We developed a simulated environment in MuJoCo for our assembly task within the imitation learning pipeline, as illustrated in Figure 4.1. The key components of the scene are as follows:

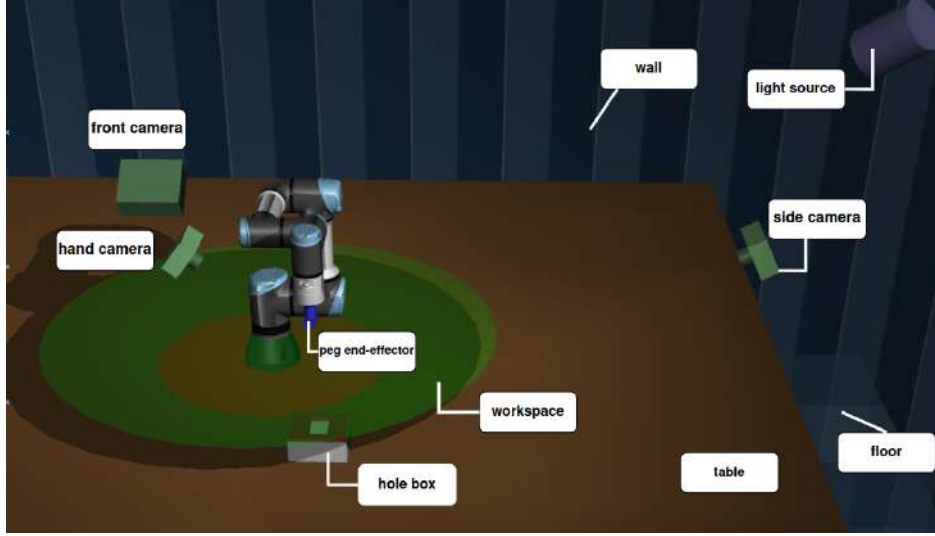


Figure 4.1: Overview of the simulation scene used in our experiments.

- **UR3e Arm:** Since the UR3e model is not directly provided, we followed the instructions from the MuJoCo Menagerie [27] to manually import and configure the model for compatibility with the `dm_control` framework. The default starting configuration is set with joint angles $[-\pi/2, -\pi/2, \pi/2, -\pi/2, -\pi/2, 0]$, as shown in Figure 4.1. This configuration serves as the reference pose when domain randomization is not applied.
- **Peg End-Effector:** A rectangular peg ($6.3 \text{ cm} \times 2.5 \text{ cm}$) is attached to the robot’s end-effector using primitive geoms. The pose of the peg is defined relative to the end-effector. To improve visual recognition, the peg is colored blue, making it more distinguishable from the environment.
- **Hole Box:** The target object consists of a rectangular box with a recessed cavity and a lid containing a hole. The hole dimensions are $6.8 \text{ cm} \times 3.0 \text{ cm}$, allowing slight clearance for insertion. The box is colored white, while the lid (which contains the hole) is marked red to emphasize the key region of interest.
- **Workspace:** The robot operates within a defined workspace bounded by two cylindrical regions (an inner and outer boundary). This area is visualized with green and red cylinders in Figure 4.1. The workspace defines where the hole box can be randomly placed and ensures reachable positions for the robot.

- **Cameras:** MuJoCo provides built-in camera instances. We utilize three views: front, side, and hand-mounted. Each camera outputs images at a resolution of 426×240 pixels, which are centrally cropped to 244×244 . The field of view (FOV) is set to 42.5 degrees.
 - **Front Camera:** Positioned at $[1.0, 0.0, 1.8]$ and oriented toward $[0.4, 0.0, 1.0]$, this view captures the entire workspace, including the hole box and the peg’s motion. It is primarily used to localize the hole position.
 - **Side Camera:** Placed at $[0.5, 1.0, 1.5]$ and looking at $[0.5, 0.0, 1.0]$, this camera provides a side view to assess height differences between the peg and the hole. It is located far and high enough to cover the workspace comprehensively.
 - **Hand Camera:** Mounted on link3 (prior to the tool link), this camera avoids tool rotation and offers a close-up view of the peg and hole during insertion. It helps with precise alignment and fine adjustments during the assembly phase.
- **Light Source:** A single directional light is placed at $[1.0, 1.0, 2.0]$, aimed toward the robot base. This enhances visual quality and consistency across the image inputs.
- **Environment Elements (Table, Walls, Floor):** Additional elements are added to emulate real-world settings, such as the robot placed on a table. These components are assigned distinct textures (as detailed in Section 3.1.3) to support domain randomization, which is essential for improving generalization. Hence, we model them as separate components.

4.3. Real Robot Deployment

This section outlines the system setup for real-world experiments with the UR3e robot, including both the control interface and the physical environment.

4.3.1. Interface with UR3e

To connect and control the UR3e robot, we use the official ROS-based driver provided by Universal Robots [28]. We follow the installation instructions in

the repository, which include setting up the URCap software on the robot and installing the ROS driver package on the server computer.

Establishing the connection between the robot and the PC is straightforward: they are connected via an Ethernet cable, and the PC's network settings are configured to match the robot's IP subnet. The ROS driver provides a launch file that initializes the controller and establishes communication with the robot. We use the 'twist_controller' to send velocity commands, and we modify the launch file parameters to load this controller by default.

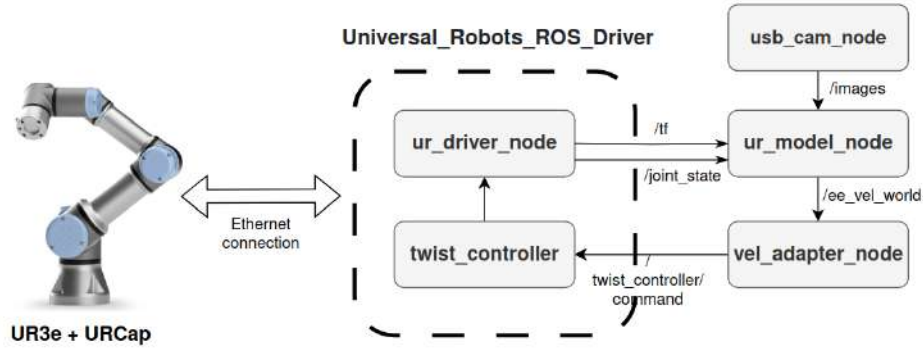


Figure 4.2: An overview of the ROS based software architecture used to interface with the UR3e robot.

For deploying our trained policy, we developed a custom ROS node named 'ur_model_node'. This node receives robot state information (joint positions and end-effector pose) from the driver node and visual input from camera drivers. These observations are then used as input to the model, which performs inference at a fixed control interval of 0.2 seconds. The resulting velocity commands are published to the controller.

However, since the 'twist_controller' expects velocity commands in the robot's base frame and our model outputs commands in the world frame, we implement an adapter node to convert between these frames. This is done by applying a fixed transformation from the world to the base frame.

An overview of the software architecture used to interface with the UR3e robot is shown in Figure 4.2.

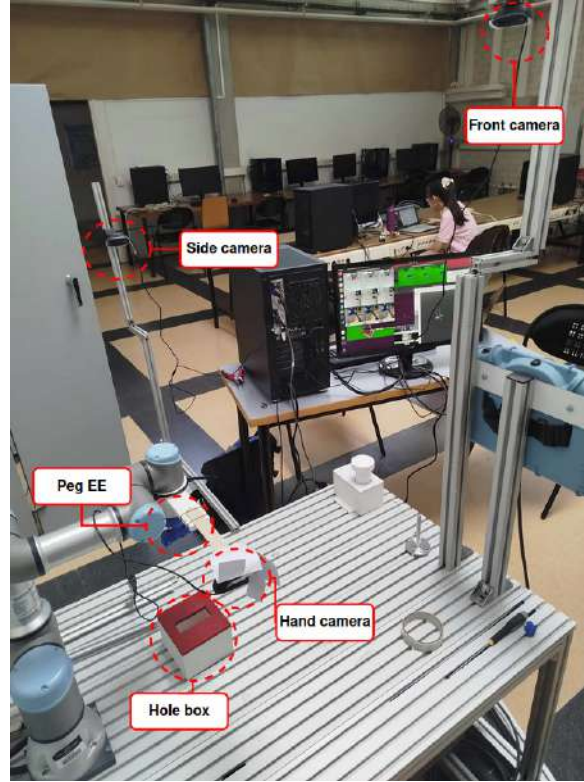


Figure 4.3: Real-world experimental setup.

4.3.2. Real Robot Setup

Figure 4.3 shows the setup used for the real-world experiments. We designed this setup to resemble the simulation environment as closely as possible. The following aspects were considered:

- **Workpieces:** We used 3D-printed components for the peg and hole box, matching the exact dimensions used in simulation. While the colors are not perfectly identical, we chose similar tones. Thanks to the domain randomization strategy applied during training (see Section 3.1.3), these small color differences should not be problematic.
- **Camera Placement:** The front and side cameras were mounted using extension beams, and the hand camera was attached with a 3D-printed holder. While we intended to place the cameras at the exact same positions as in simulation, hardware limitations made this infeasible. Instead, we manually adjusted their placements by comparing real-world camera images with simulated ones until they appeared visually aligned. Example camera views from the real setup are shown in Figure 4.5.

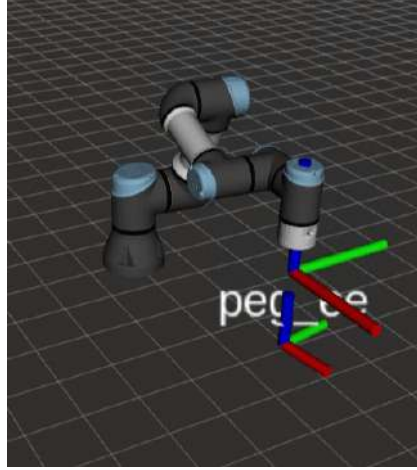


Figure 4.4: RViz visualization of the TF tree, including the static transform from the robot flange to the peg tip.

- **Robot Joint Configuration:** We attempted to set the robot’s default joint configuration to match the simulation (see Figure 4.1). However, a discrepancy was found in the joint state readings: the first joint angle (q_1) was $\pi/2$ on the real robot but $-\pi/2$ in simulation. To reconcile this, we subtract π from the received angle before passing it to the model, ensuring compatibility with the training distribution.
- **End-Effector Position:** The UR3e ROS driver provides the end-effector pose via the TF tree. Since the physical end-effector (peg) extends beyond the robot flange, we define an additional static transform to capture this offset. A TF listener retrieves the final pose of the peg, which is then used as part of the model input. An example visualization of the TF frames in RViz is shown in Figure 4.4.

4.3.3. DAGGER on the Real Robot

We also applied the DAGGER process in the real-world setting, following the same overall procedure described in Section 3.4.2, with a few notable differences in the data recollection step. In DAGGER, the expert policy is required to provide the correct action label for each observed state, which in turn depends on having access to the hole’s position and orientation. Unlike in simulation, we cannot directly obtain this information in the real world.

To address this, we manually guide the robot’s end-effector to the hole position and record it as the ground truth input for the expert policy. This manual process

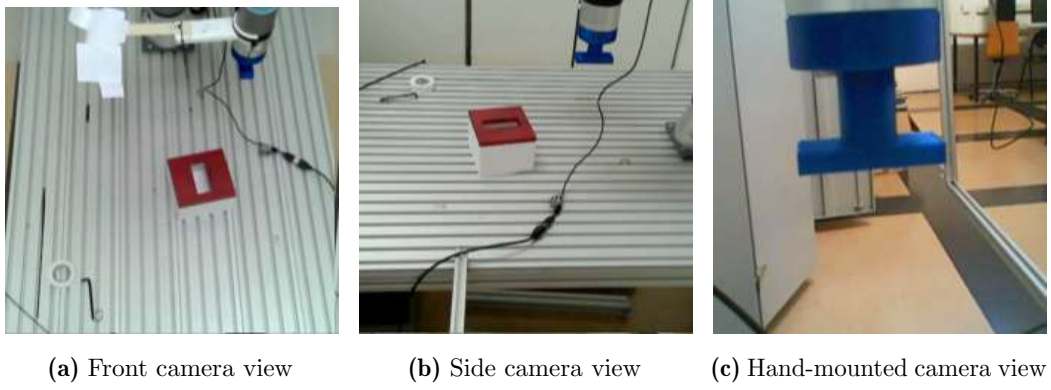


Figure 4.5: Example camera views from the real-world experiment setup.

significantly slows down data collection, taking approximately 2 minutes per demonstration.

To reduce the effort required for resetting the hole between each trial, we adopted a strategy where we fix the hole position and orientation and then record six demonstrations with varying initial peg positions. We repeat this process by subsequently altering the hole’s orientation, and then its position, ensuring diverse coverage of potential scenarios while minimizing setup time.

As described in Section 3.2.2, the time interval between input frames in simulation is approximately 1.0 second. However, this cannot be directly replicated in the real robot due to safety constraints. For safety, we run the robot at about 30% of the velocity used in simulation. To ensure that the robot’s displacement between frames visually matches that of the simulated data, we increased the interval between captured frames to approximately 3.0 seconds. Consequently, the time between individual data points during collection was adjusted to 0.6 seconds. An example of visual input data captured in the real robot setup is shown in figure 4.5.

In each iteration of DAGGER on the real robot, we collect around 50 demonstrations. These are then aggregated with the original expert demonstrations and DAGGER data from simulation. The model is retrained using the same parameters as in Table 3.1.

5. Results

This chapter presents the performance of the behavior cloning model, the improvements achieved through DAGGER, and the results of real robot deployment.

Throughout the chapter, the main reference for evaluating model performance will be the model comparison tables (Table 5.1, 5.3, 5.5). The metrics used for evaluation include the completion rate in each task state, the time taken to complete each state, and the average prediction error. The task states referenced in the tables correspond to the assembly process defined in Section 3.1.1.

The first state involves moving the peg approximately 10 cm above the hole box, positioning it in preparation for insertion. The second state is further divided into two substates to enable a more detailed analysis of the model's behavior. Substate 2.1 is considered successful when the robot positions the peg above the hole within a predefined threshold of 0.15 cm. in the xy plane and 0.1 radians. Substate 2.2 is considered complete when the peg is successfully inserted into the hole and is ready for rotation. This additional breakdown is intended to offer deeper insight into the model's performance. Example cases illustrating these substates are shown in Figure 5.1. The final state involves rotating the peg 90 degrees inside the box.

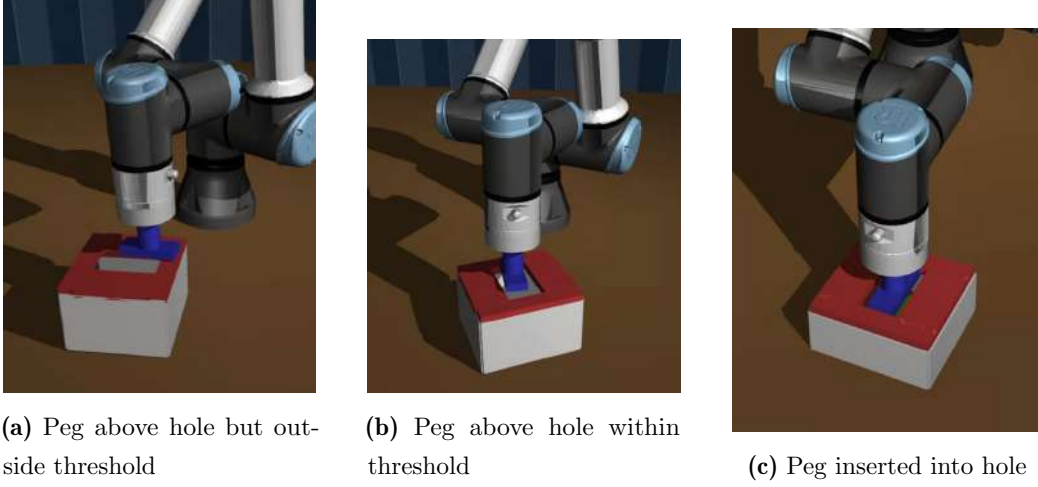


Figure 5.1: Examples of assembly substates. Subfigure (a) shows an incomplete state 2.1, (b) shows a completed state 2.1 with acceptable alignment, and (c) shows a completed state 2.2 with successful insertion.

5.1. Behavior Cloning Experiment

In this section, we evaluate several model variants in a simulation environment where only the hole position is randomized, and no other domain randomization is applied. Although many models were developed and tested, we present only those that effectively showcase the impact of the architectural modifications proposed in Section 3.3.1. Specifically, we include the vanilla model without an RNN, the vanilla model with an RNN 3.7a, the Late Fusion model without using the hole pose as an intermediate output, and the complete Late Fusion model 3.7b. These selected models form the basis of an ablation study designed to isolate the contributions of each architectural component.

Impact of 3D ResNet: Figure 5.2 presents the validation losses during training, highlighting the significant improvements achieved with the introduction of the 3D ResNet as the vision module. Compared to the two variants that rely on 2D ResNet, the 3D ResNet substantially reduces all loss components. Table 5.1 further confirms this improvement at the task level, where the model with 3D ResNet achieves a markedly higher completion rate—particularly in state 2.1, where the success rate doubles relative to the vanilla model. This improvement reflects the model’s enhanced ability to accurately localize the hole and better imitate expert motion, as evidenced by its lower velocity loss (see Table 5.2).

Impact of Temporal Modeling: When comparing the variant without any

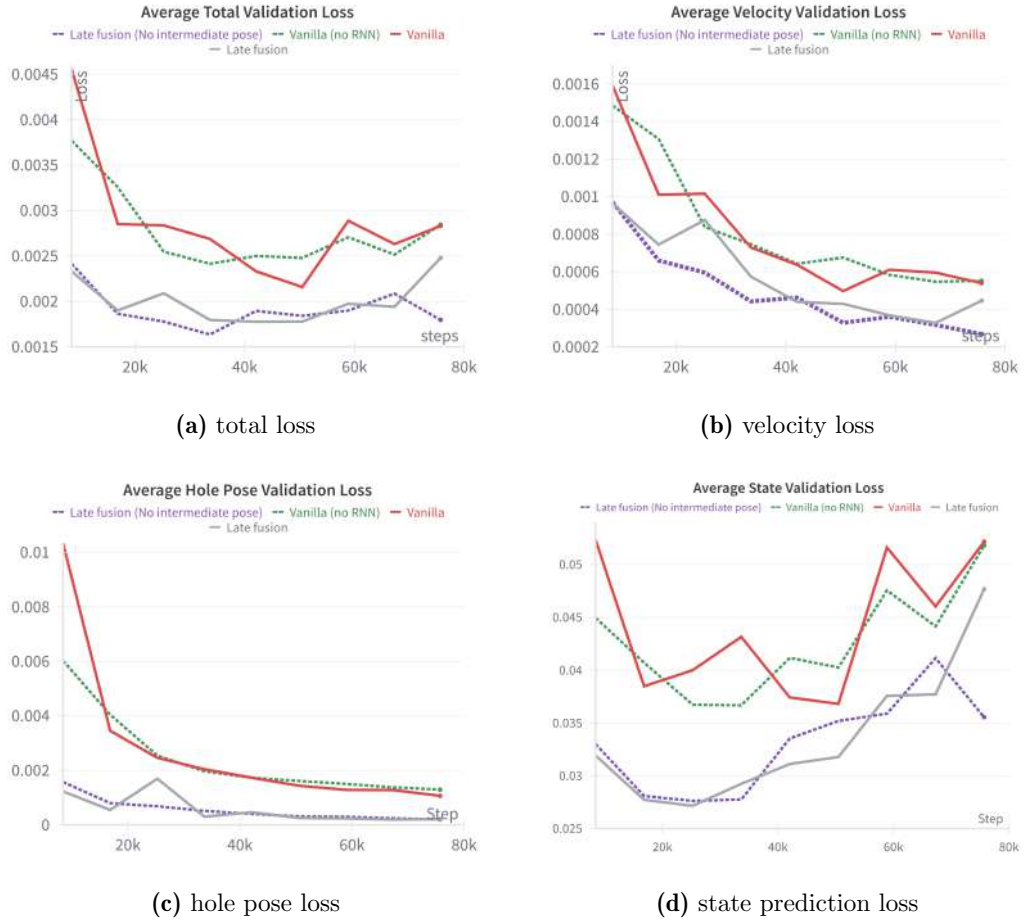


Figure 5.2: Validation loss curves of behavior cloning models across training epochs. Each subfigure compares the performance of four model variants on different loss components.

Table 5.1: Behavior cloning test: Completion Rate and Average Time per State

Model	State 1	State 2.1	State 2.2	State 3
<i>Completion Rate (%)</i>				
Vanilla (no LSTM)	95	22	3	3
Vanilla	99	36	12	12
Late Fusion (no intermediate pose)	100	74	16	16
Late Fusion	100	79	25	25
<i>Average Time (s)</i>				
Vanilla (no LSTM)	2.4	4.10	0.72	1.28
Vanilla	2.58	5.51	0.96	1.53
Late Fusion (no intermediate pose)	2.16	6.58	2.15	1.88
Late Fusion	2.38	6.20	1.30	1.65

Table 5.2: Behavior cloning test: Prediction Accuracy and Output Errors

Model	Hole. Error	Vel. Error	State Accuracy (%)
Vanilla (no LSTM)	0.143	0.201	34.51
Vanilla	0.204	0.246	45.17
Late Fusion (no intermediate pose)	0.056	0.173	47.30
Late Fusion	0.035	0.165	40.56

temporal modeling (no LSTM) to the others, it performs worst in terms of overall task completion. This performance gap appears to stem from the model’s limited understanding of task progression 5.2, as reflected in its low state prediction accuracy . The lack of temporal context likely hinders its ability to respond with the correct action to the current observation, which is essential in a sequential task like assembly.

Impact of Hole Pose as an Intermediate Output: As introduced in Section 3.3.1, we modified the architecture to explicitly predict the hole pose as an intermediate output using only the visual input (the only modality directly related to the hole). This design aims to guide the model toward better spatial understanding. Empirical results support this idea: among the 3D ResNet-based models, the variant with hole pose supervision achieves more accurate hole predictions (see Table 5.2), which translates into a higher insertion success rate. Ultimately, this enhancement contributes to a 9% increase in full task completion, underscoring its practical benefit for precise alignment tasks.

One notable observation from this experiment is the emergence of behaviors that were not explicitly demonstrated by the expert policy. In particular, when the robot fails to insert the peg into the hole during state 2, it frequently lifts the peg back to the approximate position of state 1 and repeatedly attempts insertion. This behavior is sensible in the context of the model’s training: the misaligned configurations encountered in these failure cases (as illustrated in Figure 3.8) were never present in the expert demonstrations. As a result, the model reverts to the most familiar action sequence, typically associated with state 1.

However, this fallback behavior does not contribute to improved insertion success. Despite the appearance of persistence, the model lacks the corrective strategy needed for recovery from misalignment, leading to a relatively low overall task completion rate of just 25% for the complete Late fusion model.

This limitation highlights the need to expose the model to off-distribution states—those it would encounter during real-world execution—and to guide it with more appropriate actions in those cases. To address this, we turn to the DAGGER algorithm, which we describe in the following section.

5.2. DAGGER Experiment

In this section, 5 iteration (recollecting, aggregate and retrainnig) of dagger algorithm has been applied to our complete Late fusion model. We show the

performance development and how the behavior change in each iteration compare with the original one. The experiment setup is the same as previous experiment 5.1.

Table 5.3: DAGGER test: Completion Rate and Average Time per State

Model	State 1	State 2.1	State 2.2	State 3
<i>Completion Rate (%)</i>				
Late Fusion	100	79	25	25
DAGGER Iteration#1	100	86	57	57
DAGGER Iteration#2	100	86	60	55
DAGGER Iteration#3	100	94	66	64
DAGGER Iteration#4	100	97	77	74
DAGGER Iteration#5 (Final model)	100	100	85	77
<i>Average Time (s)</i>				
Late Fusion	2.38	6.20	1.30	1.65
DAGGER Iteration#1	2.43	7.84	2.72	1.67
DAGGER Iteration#2	2.38	6.91	3.74	1.755
DAGGER Iteration#3	2.39	6.29	2.38	2.25
DAGGER Iteration#4	2.35	5.82	1.83	2.16
DAGGER Iteration#5 (Final model)	2.36	5.15	1.59	2.03

As shown in Figure 5.3a, DAGGER significantly improves the completion rate across all task states. Notably, state 2.1 reaches a 100% completion rate in the final iteration, indicating the model’s consistent ability to position the peg within the defined alignment threshold. This improvement is further supported by the reduction in both hole pose error and velocity error, as seen in Figures 5.3c and 5.3b.

Completion rates for states 2.2 and 3 see a substantial increase from 25% to 57% after just one iteration, ultimately reaching 77% by the final iteration. This jump is primarily due to the realignment behavior introduced through DAGGER training (Section 3.4.1), which enables the model to correct misalignments before attempting insertion.

The emergence of this realignment behavior is evident from the first iteration. Table 5.3 shows a significant increase in the average time to complete state 2.1,

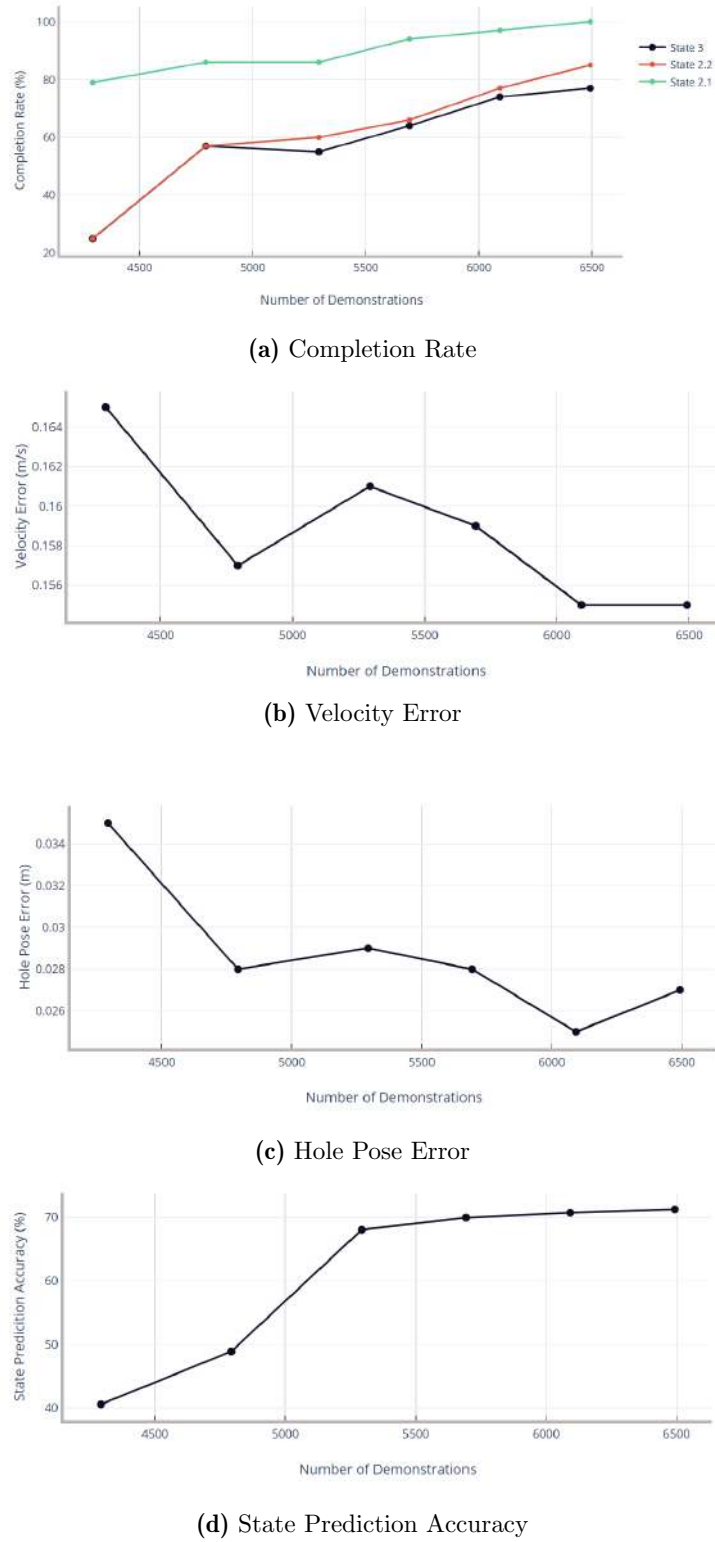
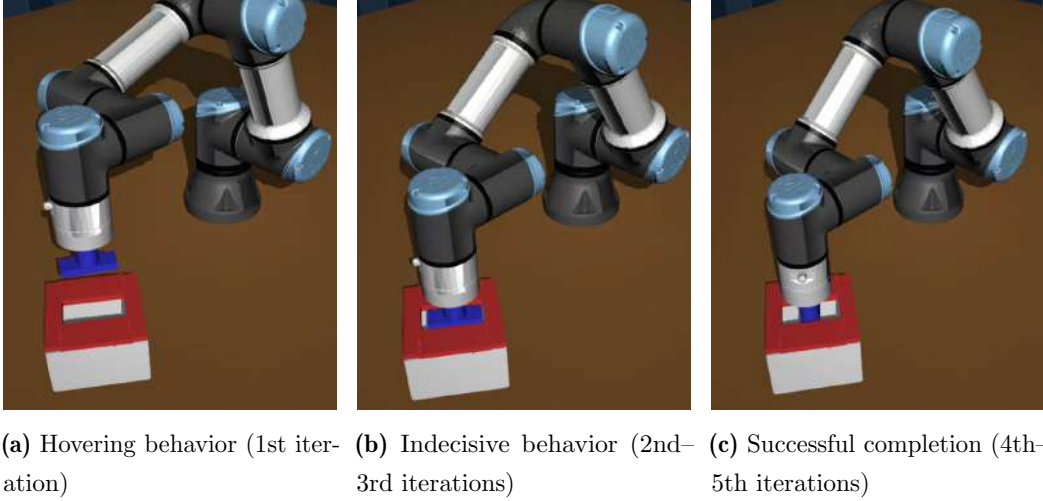


Figure 5.3: Performance progression of the DAGGER-trained model as more demonstrations are aggregated.

Table 5.4: DAGGER test: Prediction Accuracy and Output Errors

Model	Vel. Error	Hole Error	State Accuracy (%)
Late Fusion	0.035	0.165	40.56
DAGGER Iteration#1	0.028	0.157	48.90
DAGGER Iteration#2	0.029	0.161	68.04
DAGGER Iteration#3	0.028	0.159	69.96
DAGGER Iteration#4	0.025	0.155	70.74
DAGGER Iteration#5 (Final model)	0.027	0.155	71.26


Figure 5.4: Progression of robot behavior across DAGGER iterations.

reflecting the robot’s effort to fine-tune its alignment before insertion. However, during this phase, the robot often hovered above the hole without committing to the insertion, indicating excessive caution (see Figure 5.4a).

In the second iteration, this hovering behavior improved, and the model began to lower the peg more frequently. However, it still exhibited indecisiveness, occasionally withdrawing the peg even after achieving proper alignment—an issue illustrated in Figure 5.4b. This is also reflected in the increase in completion time for state 2.2 during that iteration. Over successive iterations, the model became more decisive and efficient, with average completion times for states 2.1 and 2.2 gradually decreasing. Nonetheless, hovering and indecision remained the primary failure modes in the final model, limiting its ability to reach perfect performance.

Finally, we evaluated the model’s generalization by moving the hole box

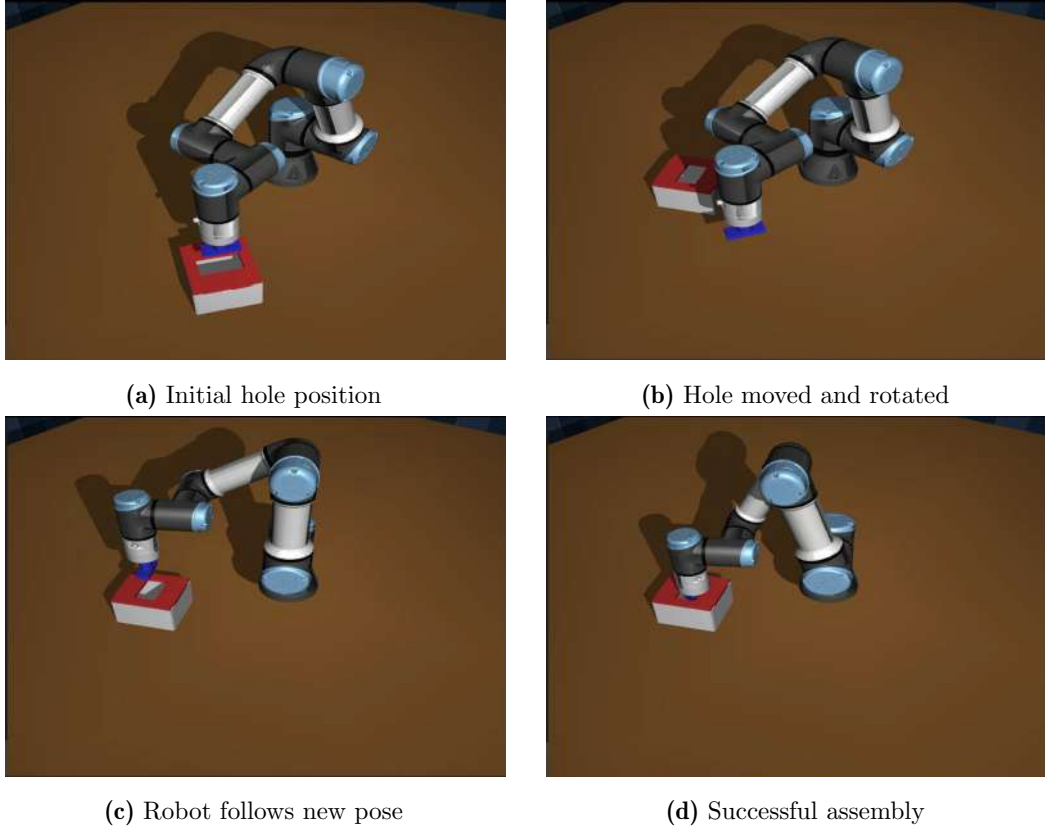


Figure 5.5: Generalization test: the robot adapts to the changing hole position and orientation.

position and orientation. As shown in Figure 5.5, the model reliably followed the box movement, attempted alignment upon reaching proximity, and proceeded with the insertion—demonstrating robust spatial awareness and adaptability, without any explicit programming in an expert to follow the box.

5.3. Real Robot Experiment

After validating the final model’s performance in simulation, we deploy it on the real robot to assess the overall effectiveness of our pipeline and evaluate the sim-to-real transferability enabled by domain randomization. Additionally, we demonstrate further improvements achieved by incorporating real-world data through two iterations of the DAGGER algorithm. The experimental setup on the real robot mirrors the simulation environment, with the only variation being the randomization of the hole position for each trial. However, due to the practical limitations of conducting real-world experiments, such as time, safety, and setup

constraints, it is not feasible to perform evaluations at the same scale as in simulation. As a result, the outcomes presented here may not fully reflect the model’s true performance potential.

Table 5.5: Real robot test: Completion Rate and Average Time per State

Model	State 1	State 2.1	State 2.2	State 3
<i>Completion Rate (%)</i>				
Final Model from simulation	20	0	0	0
DAGGER Iteration#1	100	60	0	0
DAGGER Iteration#2	100	80	0	0

Table 5.6: Real robot test: Prediction Accuracy and Output Errors

Model	Hole Error	Vel. Error
Final Model from simulation	0.17	0.320
DAGGER Iteration#1	0.08	0.26
DAGGER Iteration#2	0.06	0.22

The performance of the final model, originally trained in simulation, is reported in Tables 5.5 and 5.6. The model performs poorly in both task completion and precision. In most trials, the robot fails to reach the hole, highlighting a significant discrepancy between the simulation setup and the real-world environment, as discussed in Section 4.3.2. Additionally, the domain randomization strategy applied during training appears insufficient for bridging this sim-to-real gap—an issue we further address in the future work section.

To mitigate this, we incorporated 50 real-world demonstrations using DAGGER. This resulted in a noticeable improvement: the robot consistently moves toward the hole position, although it still struggles with the insertion phase, exhibiting the same hovering behavior observed during the early DAGGER stages in simulation.

After collecting an additional 50 demonstrations (100 in total), the robot began to descend more confidently. However, occasional misalignments still prevent successful insertions. The progression of real-world performance across iterations is visualized in Figure 5.6.

We also conducted a generalization test, similar to the one in simulation, by manually moving the hole box between trials. The robot demonstrated the

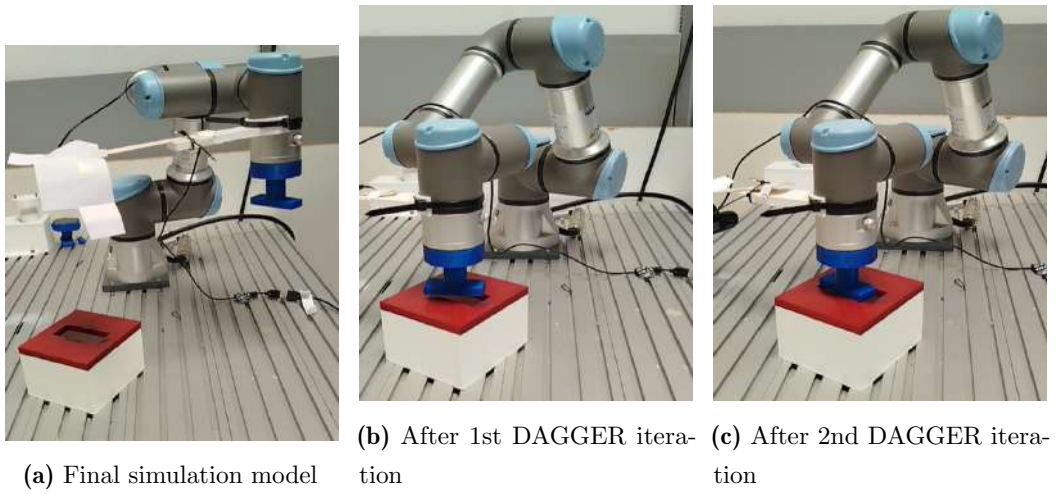


Figure 5.6: Progression of real-world performance through DAGGER iterations.

same behavior observed in simulation: it actively followed the hole position and attempted alignment and insertion. However, occasional misalignments during the assembly phase still hindered consistent task completion. Example frames from this test are shown in Figure 5.7, illustrating the robot's response to changes in hole position and orientation.

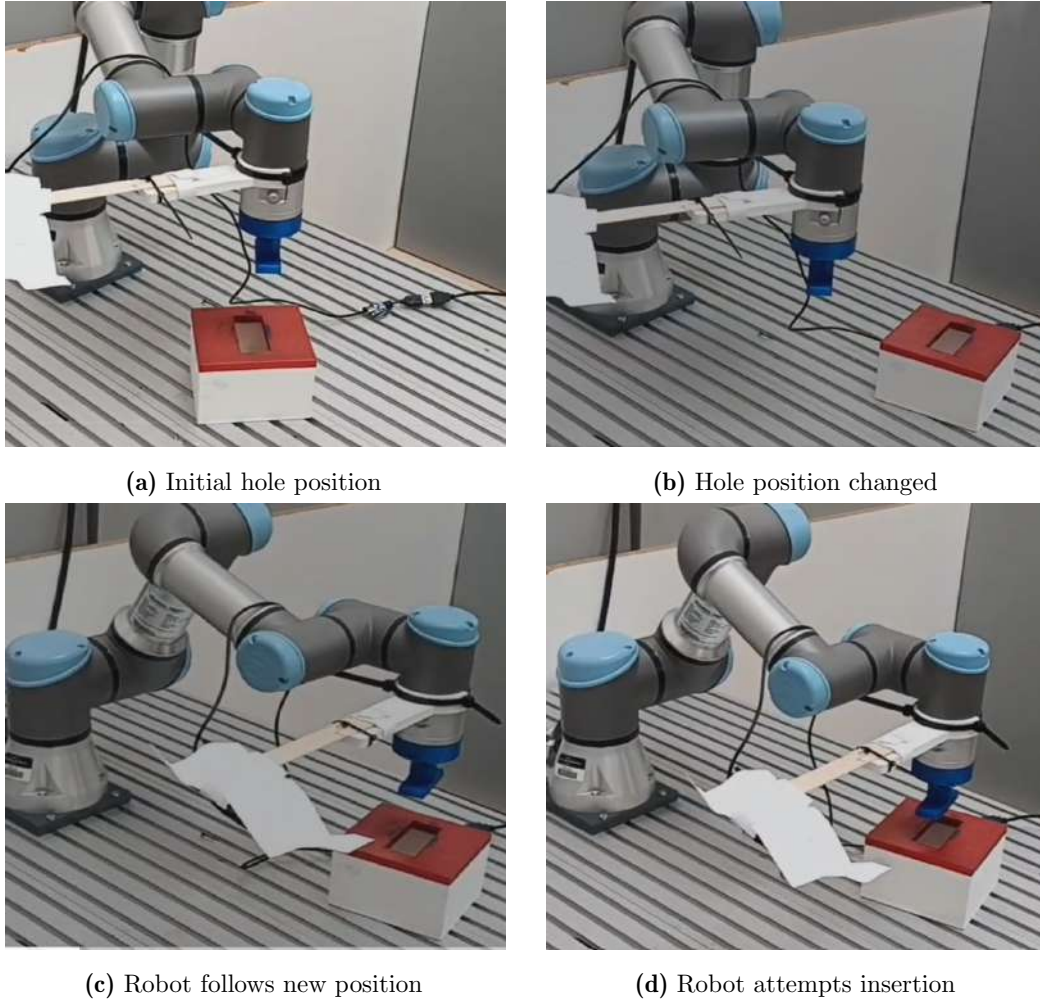


Figure 5.7: Real-world generalization test: the robot adapts to hole position and orientation changes and attempts insertion.

6. Conclusion

This thesis presents a study on imitation learning for visuomotor skill acquisition, focusing on a pipeline that leverages expert demonstrations, privileged information, and the repeatability of simulation. The proposed method enables a robot to learn an assembly task directly from visual input—without any explicit image interpretation or manual programming—by relying purely on learned behavior.

We began by designing the expert policy within a MuJoCo simulation, defining the assembly task and exploiting privileged information to generate consistent and optimal demonstrations. The simulation environment also allowed for controlled domain randomization to help bridge the sim-to-real gap. We detailed the process of data collection, dataset structure, and preparation for training.

Building on this foundation, we introduced architectural modifications to improve upon a baseline model, enhancing its ability to process visual and temporal information. Initial training was conducted using behavior cloning, after which we applied the DAGGER algorithm to further refine the model. DAGGER allowed us to correct the model’s behavior in states it had never encountered in the demonstrations, substantially improving task completion.

The model was evaluated both in simulation and on a real UR3e robot. In simulation, we conducted a comprehensive ablation study to assess the contribution of each architectural component. The results demonstrated that our proposed modifications significantly improved both precision and task completion rate. We then applied multiple iterations of the DAGGER algorithm, which exposed the model to previously unvisited states and corrected its behavior using the expert policy. As a result, the final model achieved a 78% completion rate in simulation and demonstrated the ability to generalize to varying hole positions and orientations.

When transferred to the real robot, the model initially underperformed, reflecting the insufficient calibration of our domain randomization parameters. However, after just two iterations of real-world data aggregation using DAGGER, perfor-

mance improved significantly. The model demonstrated the ability to generalize to physical variations and adapt to the real-world setup with minimal additional training.

In conclusion, this work demonstrates that visuomotor skills can be effectively learned using only data generated from simulation through supervised learning. By leveraging the repeatability, safety, and access to privileged information that simulation offers, we developed a practical pipeline for training visuomotor policies without the need for large amount of human demonstrations or hand-coded visual behavior. This approach is particularly valuable for complex tasks where real-world data collection is costly, time-consuming, or impractical.

6.1. Future Work

Improving Sim-to-Real Transfer: As discussed, the performance of the model trained in simulation significantly degraded when deployed on the real robot. This highlights two key areas for improvement:

- *Aligning Simulation with Real-World Setup:* In this work, the simulation environment was developed without direct consideration of real-world feasibility, particularly with respect to camera placement. A more effective strategy would be to first determine practical camera configurations in the real setup and then replicate those settings in the simulation to reduce the reality gap.
- *Refining Domain Randomization Parameters:* The domain randomization parameters used in this study were not thoroughly tuned. In particular, randomness in camera placement—a factor that significantly affects visual perception—was not adequately addressed. A more systematic exploration and optimization of randomization parameters could improve robustness and transferability.

Model Fine-Tuning and Optimization: Due to time constraints, there was limited opportunity to fine-tune the model or apply advanced training techniques such as model pruning or hyperparameter optimization. Future work could explore these avenues to improve performance and efficiency.

Applying the Pipeline to More Complex Tasks: This study focused on a relatively simple assembly task due to the three-month development timeline. A valuable next step would be to extend this pipeline to more complex, real-

world tasks—such as door opening or USB cable insertion—to assess whether the proposed simulation-driven approach can generalize to tasks requiring more sophisticated visuomotor coordination.

BIBLIOGRAPHY

- [1] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in *Advances in Neural Information Processing Systems*, 1989.
- [2] S. Ross, G. J. Gordon, and J. A. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” 2010.
- [3] O. Kroemer, S. Niekum, and G. Konidaris, “A review of robot learning for manipulation: Challenges, representations, and algorithms,” 2019.
- [4] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, “Learning dexterous in-hand manipulation,” *arXiv preprint arXiv:1808.00177*, 2018.
- [5] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, “Deep q-learning from demonstrations,” 2017.
- [6] T. Zhang, Z. McCarthy, O. Jow, D. Lee, D. Held, and P. Abbeel, “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation,” in *Robotics: Science and Systems (RSS)*, 2018.
- [7] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.
- [8] S. Gubbi, S. Kolathaya, and B. Amrutur, “Imitation learning for high precision peg-in-hole tasks,” in *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, Apr. 2020.
- [9] S. James, A. J. Davison, and E. Johns, “Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task,” 2017.
- [10] R. Garcia, R. Strudel, S. Chen, E. Arlaud, I. Laptev, and C. Schmid, “Robust visual sim-to-real transfer for robotic manipulation,” 2023.

- [11] R. Rahmatizadeh, X. Tang, J. Rock, P. Abbeel, and L. Bölöni, “A review on robot manipulation methods in human–robot interaction,” *Robotics and Autonomous Systems*, vol. 95, pp. 100–116, 2017.
- [12] D. Brandfonbrener, S. Levine, and T. Hashimoto, “When should we prefer offline reinforcement learning over behavioral cloning?,” *arXiv preprint arXiv:2206.04779*, 2022.
- [13] Y. Zhang and K. Cho, “Query-efficient imitation learning for end-to-end autonomous driving,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.
- [14] W. Sun, A. Venkatraman, B. Boots, and J. A. Bagnell, “Deeply aggravated: Differentiable imitation learning for sequential prediction,” in *Proceedings of the 34th International Conference on Machine Learning*, pp. 3309–3318, 2017.
- [15] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, “One-shot visual imitation learning via meta-learning,” in *Conference on Robot Learning*, pp. 357–368, 2017.
- [16] Y. Zhu and et al., “Deep visual foresight for planning robot motion,” in *IEEE Transactions on Robotics*, 2022.
- [17] N. M. Shafiullah, A. Jain, F. Xu, J. Tompson, and S. Levine, “Clip-fields: Weakly supervised semantic fields for robotic memory,” in *Conference on Robot Learning (CoRL)*, 2022.
- [18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, IEEE, 2018.
- [19] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, IEEE, 2017.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CVPR*, 2016.

- [21] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6450–6459, 2018.
- [22] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, “Describing textures in the wild,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3606–3613, 2014.
- [23] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.
- [24] “Manipulator-mujoco.” <https://github.com/ian-chuang/Manipulator-Mujoco.git>. Accessed: 2025-06-07.
- [25] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa, “dm_control: Software and tasks for continuous control,” *Software Impacts*, vol. 6, p. 100022, 2020.
- [26] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, *et al.*, “Gymnasium: A standard interface for reinforcement learning environments,” *arXiv preprint arXiv:2407.17032*, 2024.
- [27] K. Zakka, Y. Tassa, and MuJoCo Menagerie Contributors, “MuJoCo Menagerie: A collection of high-quality simulation models for MuJoCo,” 2022.
- [28] U. Robots, “Universal robots ros driver.” https://github.com/UniversalRobots/Universal_Robots_ROS_Driver, 2020. Accessed: 2025-06-08.

Abstract

Robots often struggle with complex visuomotor tasks due to the difficulty of interpreting raw visual input and transferring learned control policies from simulation to the real world. This thesis explores imitation learning as a solution, using expert demonstrations generated entirely in simulation to train end-to-end visuomotor policies without explicit visual programming. We developed an expert velocity-based controller with access to ground-truth information to perform an assembly task, creating a high-quality dataset for training. The proposed model architecture leverages a 3D ResNet as the vision backbone to effectively capture temporal features. Initial training is performed via behavior cloning (BC), followed by iterative refinement using the DAGGER algorithm to expose the model to previously unseen states and improve robustness. Finally, we deploy the trained model on a real UR3e robot to evaluate its sim-to-real performance. Our approach achieved a 78% task completion rate in simulation, and although real-world performance initially lagged, it improved significantly with the integration of a small amount of real-world data.

Keywords: Imitation Learning; Behavior Cloning; DAGGER; Simulation-to-Real Transfer

15

**Vision-Based Tracking and Following of a Moving Target Using an
Unmanned Aerial Vehicle
Fatima Yousif Rustamani**

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING

MASTER THESIS

**Vision-Based Tracking and Following
of a Moving Target Using an
Unmanned Aerial Vehicle**

Fatima Yousif Rustamani

Spain, June 2025.

ACKNOWLEDGEMENTS

I address my sincere gratitude to all of those with whom I had the chance to meet and cooperate with at Saxion SMART Mechatronics and Robotics Research Group, Netherlands, and University of Zagreb, Faculty of Engineering (FER) and University of Girona under the Intelligent Field Robotic Systems (IFROS) program. Every environment from which I traversed during this trajectory had been an opportunity to reinvent myself and gain in maturity through the renewed challenges and concepts to which I was confronted. Special thanks to my supervisors, Benjamin Van Manen, for providing me the opportunity to work on this project at the SMART Research Group, alongside Stjepan Bogdan, from whom I could greatly benefit and expertise in aerial robotics. Overall, I should bestow my acknowledgment to my supervisors for discovering and polishing my inclination towards scientific research and innovation. I would also like to express my sincere gratitude to Kousheek Chakraborty, and Chandan Kumar Rajendra from SMART and Filip Zorich, and Marijana Peti from FER, for their invaluable support in helping me navigate technical challenges and for their guidance. Finally, I am deeply thankful to my parents, siblings, and friends, whose unwavering support, encouragement, and love helped me overcome every challenge along the way.

ABSTRACT

Vision-Based Tracking and Following of a Moving Target Using an Unmanned Aerial Vehicle

This work introduces an autonomous system for mobile target tracking and following using vision-based (RGB) uni-modal data, specifically designed for unmanned aerial vehicles (UAVs) and enhanced by multi-target information. It addresses the gap in current research by applying state-of-the-art multi-object tracking (MOT) techniques to target following scenarios, moving beyond traditional single-object tracking (SOT) methods. The system combines the real-time object detector YOLOv8 with MOT algorithms BoT-SORT and ByteTrack to extract and utilize multi-target data, improving re-identification performance and reducing ID switches, especially under partial or full occlusions in dynamic environments. A 3D flight control mechanism is implemented to enable responsive target following, maintaining line-of-sight despite changes in target speed or direction. The system is validated through simulation testing, demonstrating accurate and robust tracking that effectively differentiates the intended target from surrounding bystanders. By tackling key challenges, this work paves the way for practical UAV applications in vision-based target following using multi-target information.

Keywords: Autonomous UAV tracking, machine vision, unmanned aerial vehicles, object detection, multi-object tracking

CONTENTS

Acknowledgements	i
Abstract	ii
List of Figures	iv
1. Introduction	1
1.1. Topic Overview	1
1.2. Problem Statement	2
1.3. Motivation	3
1.4. Research Questions	4
1.5. Thesis Outline	4
2. Literature Overview	5
2.1. SOT and MOT	5
2.2. Object Detectors and Trackers	6
2.3. Distance Estimation and Flight Controller	7
2.4. Existing System	8
3. Methodology	9
3.1. Visual Detection and Tracking Module	11
3.1.1. Object Detection and Tracking	11
3.1.2. Target Acquisition and State Estimation	12

3.1.3. Redetection Algorithm	15
3.2. Distance Estimation	20
3.3. Flight Controller	21
3.4. System Mode Switcher	25
4. Experiments and Results	28
4.1. Performance Metrics	28
4.2. Simulation Setup	29
4.2.1. Redetection Results	29
4.2.2. Results and Analysis	34
5. Conclusion	41
5.1. Future Work	41
6. References	44

LIST OF FIGURES

3.1. System Architecture	10
3.2. Simulation detections	12
3.3. Flowchart representation of target ID change mode	16
3.4. Flowchart representation of target missing mode	17
3.5. Flowchart representation of target lost mode	18
3.6. IOU representation	19
3.7. UAV-Target reference frames and relative distance d to the target	21
3.8. Simulink: UAV highlevel controller	25
3.9. Diagram of system operation modes.	26
4.1. Target ID Change redetection experiment with another bystander: system recognizes and adapts to ID changes from the MOT module by updating the locally defined target ID. (a) to (c) Target followed with ID 47. (d) Occlusion due to bystander. (e) Target ID Change overwrite local target ID from 47 to 115. (f) Target followed with new ID	30

4.2. Target Missing redetection experiment: the target is redetected after temporary occlusion when crossing behind a bystander. (a) to (c) Following the target, Bystanders in the left. (d) Target crosses behind the bystander. Initiate Target Missing Mode. Target Missing, following Kalman filter estimate. (e) and (f) Target redetected and followed.	31
4.3. Target Lost Experiment—obstacle occlusion redetection scenario. Target successful redetection after full occlusion behind fountain in the simulation environment (a non-identifiable object). (a) Target Following Mode with ID 1. (b) Target hidden behind fountain. (c) and (d) Target lost. Stop movement. (d) Target reappears. (e) Target redetected with ID 14. Adjusting Mode. (f) Continue Target Following Mode.	33
4.4. Spatial heatmap showing the trajectory of the mobile target’s centre within the image frame during single experiment. This visualization highlights how the target was consistently tracked and kept within view during the target-following process, showing the areas of highest occurrence over time.	37
4.5. Drone and Target 2D (X-Y) Trajectories	38
4.6. Estimated distance and respective error (distance estimation error) over time for one of the BoT-SORT experiment.	39
4.7. Tracking error of the drone controller	40

1. Introduction

1.1. Topic Overview

The use of Unmanned Aerial Vehicles (UAVs) for autonomous target tracking has gained significant traction in both academic and industrial research due to its broad range of applications, including airborne surveillance [1], industrial monitoring [2], vehicle tracking [3], and automated search and rescue [4]. In this thesis, we focus on the specific task of tracking and following a moving human target using an onboard vision system. This task poses a number of challenges, particularly in complex and dynamic environments that may involve multiple moving objects, occlusions, and variations in lighting and scale.

A key aspect of ground-based target tracking using UAVs is the need to maintain a consistent line-of-sight (LOS) with the target while the drone is in motion. Whether the target is static, moving slowly, or maneuvering unpredictably, the UAV must adapt its flight path and camera orientation accordingly. Traditionally, single-object tracking (SOT) methods such as Kalman filtering or Kernelized Correlation Filters (KCF) have been employed for such tasks. However, these approaches often rely on fixed motion models and struggle under conditions of target occlusion, appearance changes, and identity switching, particularly in multi-target scenarios.

Recent developments in real-time computer vision have introduced advanced detection and multi-object tracking (MOT) frameworks that offer improved ac-

curacy and robustness. In particular, object detectors in the YOLO (You Only Look Once) family, such as the YOLOv8, have demonstrated state-of-the-art performance in terms of detection speed and accuracy. Similarly, MOT algorithms like ByteTrack and BoT-SORT extend traditional tracking by leveraging deep appearance features and association metrics to handle challenges like ID switching, re-identification, and occlusion recovery. These systems have proven effective in high-frame-rate applications, making them well-suited for UAV-based operations where latency and responsiveness are critical.

Given these advances, this feasibility study proposes a UAV-based target tracking system that integrates unimodal 3-channel RGB (vision-based) input with real-time object detection and tracking frameworks. By combining YOLOv8 for detection and MOT algorithms such as ByteTrack and BoT-SORT for tracking, the system aims to achieve reliable and persistent tracking in dynamic environments. The architecture also includes a 3D flight control loop (covering horizontal, vertical, and yaw movements) that allows the UAV to continuously follow the target while adapting to its changing position in real-world space.

1.2. Problem Statement

In this work, the problem of vision-based mobile target tracking (MTT) and following that involves several interconnected challenges is solved. Initially, the UAV must detect the target in real time using RGB image data. Following detection, a tracking algorithm monitors the position of the target within the image frame over time. To enable the UAV to physically follow the target, it must compute the position of the target relative to itself and translate that into control commands for UAV navigation.

Traditional approaches, such as single-object tracking (SOT) using KCF and Kalman filtering, often struggle in complex scenes involving occlusions and sudden

direction changes as they rely on simple consistent motion models to track the target. In addition, many systems require manual target initialization and lack robustness in real-time redetection. These issues limit practical deployment in real-world environments. As UAVs move through scenes with multiple individuals or objects, distinguishing and maintaining persistent identity of the designated target becomes increasingly difficult. Hence, more advanced approaches using multi-object tracking (MOT), recovery after occlusion are needed to overcome these limitations and support continuous and reliable following behavior. This is further discussed in the subsequent literature review section.

1.3. Motivation

Object recognition, localization, and tracking play an essential role in computer vision applications. Developing an autonomous system for robust mobile target tracking has numerous real-world applications and addresses critical gaps in current solutions. However, particularly as this thesis assignment is under the "Avoiding the Invisible" project hence a key motivation for this research comes from the fact that security systems often trigger false alarms about 80% due to non-threatening movements such as birds or wind. These false alerts require manual verification, leading to utilising human resources. Deploying an autonomous UAV in such scenarios can reduce the need for human intervention and make the system more efficient. Beyond this, the research has broader applications in areas like intrusion detection in train yards, industrial site surveillance, and supporting police or search and rescue operations, where accurate and real-time target tracking can significantly enhance situational awareness and response.

1.4. Research Questions

The core focus of this research is to explore how vision-based target tracking and following can be developed and implemented using a UAV. To achieve this, the following research questions must be addressed, as they will shape the design of the proposed tracking and following algorithm and its testing in real-world environments:

- Can MOT algorithms (BoT-SORT and ByteTrack) paired with YOLOv8 outperform SOT methods for UAV target tracking?
- Can multi-target data improve occlusion recovery?
- Does our flight controller ensure stable following in dynamic scenes?

1.5. Thesis Outline

The remainder of the dissertation is organised as follows. In Chapter 2 the related work is presented and in Chapter 3 an overview of the different stages of the tracking and following process is given along with a detailed explanation of the visual trackers and the control system design. In Chapter 4 we cover the results of simulation experiments with quantitative and qualitative results. Finally, Chapter 5 concludes the thesis with the research findings and proposes future directions for this work.

2. Literature Overview

2.1. SOT and MOT

At present, target tracking in images is generally approached using one of two strategies: Single-Object Tracking (SOT), which focuses on detecting and following a single designated target, and Multi-Object Tracking (MOT), which involves detecting and tracking multiple targets simultaneously, assigning a unique ID to each.

Numerous studies [5, 6, 7, 8, 9] have explored these approaches. Among them, SOT methods, particularly those based on the Kernelized Correlation Filter (KCF) [10], have gained wide adoption. KCF and other correlation-based trackers utilize one-shot learning and deliver reliable performance without requiring GPU support, making them ideal for resource-constrained embedded platforms [11].

However, KCF depends heavily on the visual appearance of the target. Changes due to occlusions or significant alterations in the target's appearance can disrupt tracking and lead to failure. Moreover, KCF typically requires manual initialization, as the target must be specified in the first frame [5, 6, 9]. To address the challenge of re-identifying the target after occlusion, some works (e.g., [6]) have proposed analyzing inter-frame motion to detect target-like movements. Yet, these methods often struggle in noisy or dynamic environments. An alternative and more robust solution is to combine KCF with the Kalman Filter (KF), which has been explored in several studies [8, 9].

2.2. Object Detectors and Trackers

Deep learning-based (also single-stage) object detectors such as YOLOv3 can be used to initialize KCF trackers and re-detect targets following full occlusions—particularly when the scene lacks visually similar objects [8]. In contrast, MOT methods are more robust in dynamic multi-target environments but often face challenges related to camera motion and perspective shifts introduced by UAV movement. Recent research efforts have sought to address this issue by refining camera motion models [12, 13]. MOT systems generally operate under a tracking-by-detection paradigm, where each frame involves both detection and tracking steps. This contrasts with traditional SOT approaches, which typically detect the target only once during initialization. One key benefit of the MOT paradigm is its compatibility with newer, more powerful detection models such as YOLOv8 [14] - to enhance system reliability and performance.

The effectiveness of such tracking-by-detection pipelines has been demonstrated using YOLOv7 and the state-of-the-art BoT-SORT tracker, showing strong resilience to occlusions and identity switches in pedestrian tracking tasks, even under low-light conditions and complex backgrounds [15]. Furthermore, similar architectures integrating YOLOv8 and BoT-SORT in a synthetic aperture radar imaging framework have also proven capable of real-time, high-precision tracking and detection [16]. Despite these advances, the use of MOT techniques in real-world UAV-based target following scenarios remains limited. This thesis aims to fill that gap by demonstrating the viability of MOT for UAV applications, emphasizing its strengths in handling dynamic environments and its potential to overcome key SOT limitations. Notably, the incorporation of multi-target awareness introduces capabilities such as flexible target switching, crowd following, and robust redetection mechanisms.

YOLOv8 [17] was selected for this study due to its recent release by Ultralytics — the creators of the widely adopted YOLOv5. As the YOLO family continues to

evolve, further improvements have already emerged with YOLOv9 [18], YOLOv10 [19], and the most recent YOLOv11 [20], indicating a vibrant and fast-paced research domain.

2.3. Distance Estimation and Flight Controller

In addition to detection and tracking, accurately estimating the target's position relative to the UAV is critical for effective target following. The discrepancy between the target's pixel position and the centre of the image frame is often used to guide the UAV's camera or gimbal toward achieving line-of-sight (LOS) with the target [5, 6, 7, 8]. Several studies leverage this deviation to adjust the UAV's orientation, either by steering a gimbal [5, 6] or through lateral and vertical movements to maintain the target in focus [7, 8]. Once LOS is established, maintaining a consistent distance between the UAV and the target becomes crucial. Distance estimation techniques, such as those based on the standard pinhole imaging model combined with extended Kalman filters (EKF), help manage observation noise and improve accuracy in dynamic scenarios [6]. Moreover, some methods adjust the UAV's following speed by calculating the proportion of the target's size in the image, providing another mechanism for precise target following [5, 8]. Control algorithms for mobile target following vary across studies, from direct 3D position inputs [17, 18] to the use of proportional navigation (PN) with cascade PID controllers [5]. Other works use a switchable tracking strategy based on estimated distance, transitioning between observing and following modes with different control methods [6]. Many existing distance estimation methods suffer from limitations in precise target positioning, highlighting the need for more accurate methods.

2.4. Existing System

A closely related study [21] presents a vision-based system that leverages multi-target information for robust tracking and following of mobile targets. It performs a comparative analysis of the state-of-the-art YOLOv8 object detection algorithm in combination with leading MOT methods - BoT-SORT and ByteTrack to evaluate their effectiveness in target-following scenarios. This work also introduces a 3D flight control algorithm that utilizes RGB-D data to accurately follow designated targets. Furthermore, it addresses challenges such as ID switches and partial or full occlusions in dynamic environments by proposing a novel redetection algorithm that capitalizes on the strengths of multi-target data. The present research builds upon this existing system, however for the distance estimation, RGB input is only used without the depth component. Its feasibility is demonstrated through simulation using both ROS1 and ROS2 with the PX4 x500 UAV platform.

3. Methodology

The system architecture is shown in Figure 3.1. The overall system can be split in to four following modules:

1. Visual detection and tracking module
2. Distance estimation
3. Following flight controller
4. System mode switcher

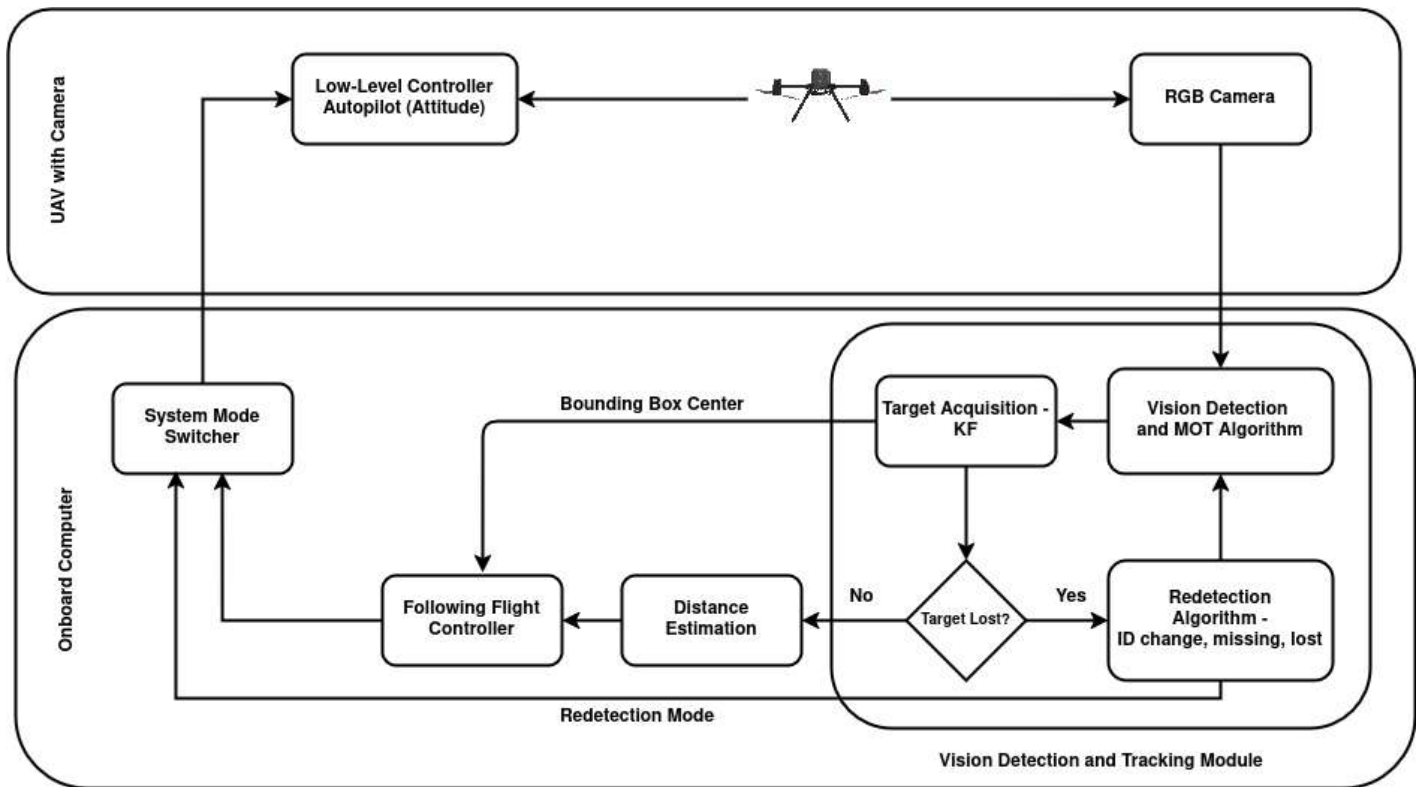


Figure 3.1: System Architecture

The “vision detection and MOT module” is responsible for detecting and tracking all objects of interest within the UAV’s field of view, including the designated target to follow. This module integrates three main components: the vision detection and MOT algorithm, which detects and tracks all objects in the image; the target acquisition stage, where a Kalman filter is employed to estimate the position of the selected target; and the redetection algorithm, which monitors the status of the target and attempts to redetect it in the case of occlusion or loss. The module processes RGB data from the UAV’s camera to differentiate between the primary target and other objects, such as bystanders, even during occlusions. When the redetection algorithm is triggered, a “redetection mode” signal is sent to the system mode switcher, which adjusts the control output accordingly.

The bounding box data produced by the vision detection and MOT module is then passed to the distance estimation stage, where the relative distance between

the UAV and the target is calculated. Based on the position of the target relative to the image centre as the estimated distance, the controller computes the necessary yaw rate and velocity commands for the UAV. These commands are then relayed to the system mode switcher, which decides the appropriate inputs to send to the autopilot, depending on the current mode of the system.

This system was tested in complex scenarios involving multiple moving pedestrians, where challenges such as occlusions and disturbances are likely to arise. However, the system is designed to be versatile and can be adapted for different objects of interest (besides human/person) by modifying the training dataset used for the object detector and adjusting the control parameters as needed. Additionally, alternative distance estimation methods can be explored and applied to other object types, ensuring the system's effectiveness across a variety of applications

3.1. Visual Detection and Tracking Module

The first module is the visual detection and tracking module responsible for processing the images from the camera, identifying the target and performing necessary redetections.

3.1.1. Object Detection and Tracking

The first step in achieving target following involves detecting the target as well as any bystanders in the image and assigning each a unique identifier (ID) for continuous tracking. This is performed using a track-by-detection approach, where a real-time object detector identifies and classifies objects in each frame for subsequent tracking.

For the detections, the system takes advantage of the state-of-the-art detector YOLOv8. The YOLOv8 is pre-trained on the COCO dataset, which includes 80 object classes. However, for the purposes of target following, only detections

corresponding to the “person” class are retained. The YOLOv8 model will analyse the image in real-time and provide a list of detections, highlighting all the objects in the image that are identified with the person class above a threshold confidence level. These detections contain: bounding box surrounding the object, class of the object and confidence score in that result.

The `ultralytics_ros` package makes the detection and tracking easy to be implemented in ROS by providing the detection results topic. An example of a detection from simulation scenario is shown in the Figure 3.2.

The detections (composed of the bounding boxes and classes) are then linked over time by the tracker, by attributing unique identifiers (IDs) for each object, as also shown in Figure 3.2. Two state-of-the-art trackers are considered: the BoT-SORT MOT and its predecessor, ByteTrack. YOLOv8 provides the detections via bounding boxes to the tracker which in turn performs data association with the previous frame to match each detection with a corresponding ID.



Figure 3.2: Simulation detections

3.1.2. Target Acquisition and State Estimation

After the take-off and initialization of the detection and tracking algorithm, users can select a target to follow from the list of detected individuals. In this work, the

algorithm automatically assigns the first detected person as the target to follow, storing their respective ID for tracking in subsequent frames. To fully leverage the capabilities of MOT, the positions of other detected individuals and their assigned IDs are also recorded to improve redetection in case the target is occluded.

After detection, a Kalman filter is used to predict the target's position even during occlusions [9]. The prediction model employed is based on a constant velocity motion model [22], where the velocities are derived from the movement of the target's bounding box centre in the image. The 6D state used includes the centre coordinates of the bounding box (C_x, C_y), the size of the bounding box (width - w, height - h) and the speed of movement in the image ($\dot{C}_x, \dot{C}_y$), calculated from the movement of the centre of the bounding box pixel coordinates. The state and the 4D observation vector are defined in Equation 3.1.

$$\mathbf{x}(t) = \begin{bmatrix} C_x \\ C_y \\ \dot{C}_x \\ \dot{C}_y \\ w \\ h \end{bmatrix} \quad \text{and} \quad \mathbf{z}(t) = \begin{bmatrix} C_x \\ C_y \\ w \\ h \end{bmatrix} \quad (3.1)$$

The state equation of the system to be estimated is as:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{w}_k$$

The observation equation is:

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

Where, $\mathbf{x}_k$ is the state value of the system at time k , $\mathbf{z}_k$ is the measurement value at time k , $\mathbf{A}_k$ is the state transition matrix of the system, $\mathbf{w}_k$ is the system

error, $\mathbf{H}_k$ is the measurement matrix, and $\mathbf{v}_k$ is the measurement error. Both $\mathbf{w}_k$ and $\mathbf{v}_k$ are noise terms conforming to a Gaussian distribution. The iterative process of the Kalman filter is divided into two parts: prediction and update.

while the prediction equations are as:

$$\begin{aligned}\mathbf{x}_k^- &= \mathbf{F}\mathbf{x}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^\top + \mathbf{Q}\end{aligned}$$

Here, $\mathbf{x}_k^-$ is the predicted (a priori) state estimate at time k , and $\mathbf{P}_k^-$ is the predicted covariance matrix. $\mathbf{F}$ is the state transition model, and $\mathbf{Q}$ is the process noise covariance.

and the update equations are as:

$$\begin{aligned}\text{Residual: } \mathbf{y}_k &= \mathbf{z}_k - \mathbf{H}\mathbf{x}_k^- \\ \text{Innovation covariance: } \mathbf{S}_k &= \mathbf{H}\mathbf{P}_k^-\mathbf{H}^\top + \mathbf{R} \\ \text{Kalman Gain: } \mathbf{K}_k &= \mathbf{P}_k^-\mathbf{H}^\top\mathbf{S}_k^{-1} \\ \text{Updated state: } \mathbf{x}_k^+ &= \mathbf{x}_k^- + \mathbf{K}_k\mathbf{y}_k \\ \text{Updated covariance: } \mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H})^\top + \mathbf{K}_k\mathbf{R}\mathbf{K}_k^\top\end{aligned}$$

The other variables in the Kalman filter formulation namely the state transition matrix $\mathbf{F}$, the observation matrix $\mathbf{H}$, the process noise covariance $\mathbf{Q}$, the measurement noise covariance $\mathbf{R}$, and the error covariance matrix $\mathbf{P}$ are initialized and tuned based on empirical observations. Specifically, $\mathbf{F}$ is designed to reflect the constant velocity model by incorporating the time step Δt between frames, while $\mathbf{H}$ is structured to extract only the observable elements of the state vector.

The matrices $\mathbf{Q}$ and $\mathbf{R}$ were manually tuned based on a of simulated experiments to balance responsiveness with stability. The error covariance matrix $\mathbf{P}$ is initialized with moderate uncertainty and dynamically updated as the filter runs.

Note: The filterpy [23] library was used in this implementation.

3.1.3. Redetection Algorithm

In vision-based detection and tracking of mobile targets from UAVs, occlusions pose a significant challenge, especially in densely populated areas. Redetection algorithms are crucial for reliable operations, particularly in scenarios involving multiple individuals. In such systems, the availability of multi-target information is a valuable asset to enhance the redetection algorithms and prevent the loss of the target.

Starting from the beginning of the target following process, the algorithm verifies detections using the assigned IDs from the YOLOv8+BoT-SORT or YOLOv8+ByteTrack setups by matching the received detections IDs with the locally stored detection list IDs. If the assigned target ID is absent in the detections, three possible situations may arise:

1. **Target ID Change:** A limitation of MOT algorithms that occurs when the target is still visible in the image; however, the MOT algorithm assigned it a different identity than the previous frame. This may happen due to sudden movements of the target or the UAV that cause the tracker to misidentify the target. Since the algorithm identifies the target via the assigned ID, an identity change would cause a system failure if left unattended. Hence, it is necessary to update the locally defined target ID with the new assigned ID from the MOT module for continuous target following. The redetection algorithm allows the system to keep accurate target tracking and following, despite the MOT limitations. The flowchart in Figure 3.3 shows the target ID change mode handling.

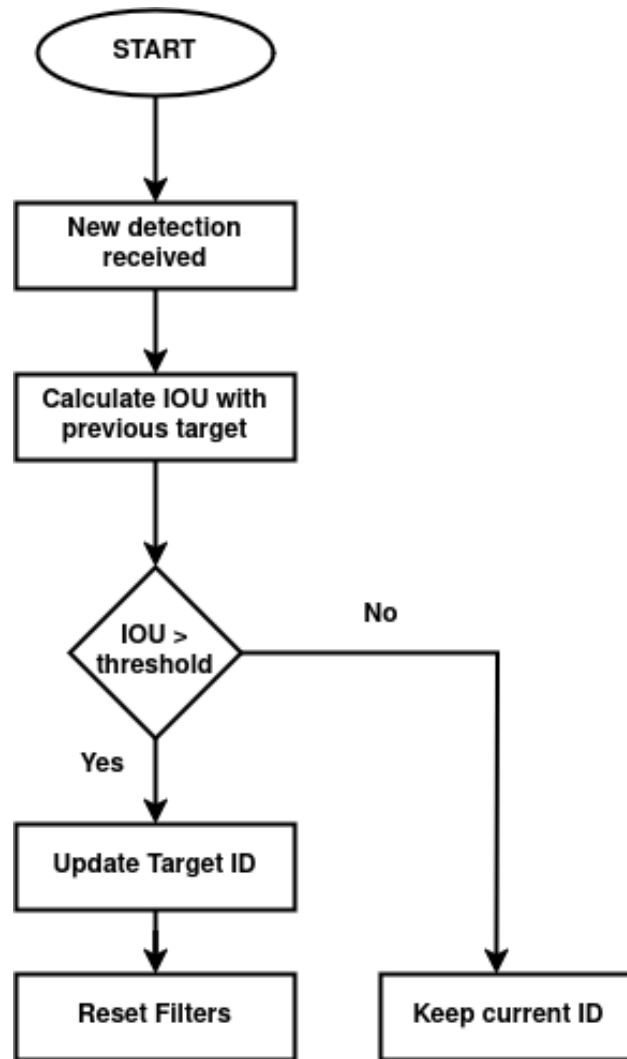


Figure 3.3: Flowchart representation of target ID change mode

2. **Target Missing:** Initiates when the target is no longer detected in the image, potentially due to occlusions or obstructions by other objects, and a counter is initiated. During this step, the system continues to follow an estimate of the position of the target given by the Kalman filter. The brief window where the Kalman filter is used allows for a more robust system that will not immediately stop for short occlusions, like when two people cross paths. The Missing stage also adds some robustness to fast target movements that would cause the target to leave the image, by continuing the corrective maneuver even after the target leaves the field-of-view. In the Figure ?? the

flowchart is shown for the target missing mode handling.

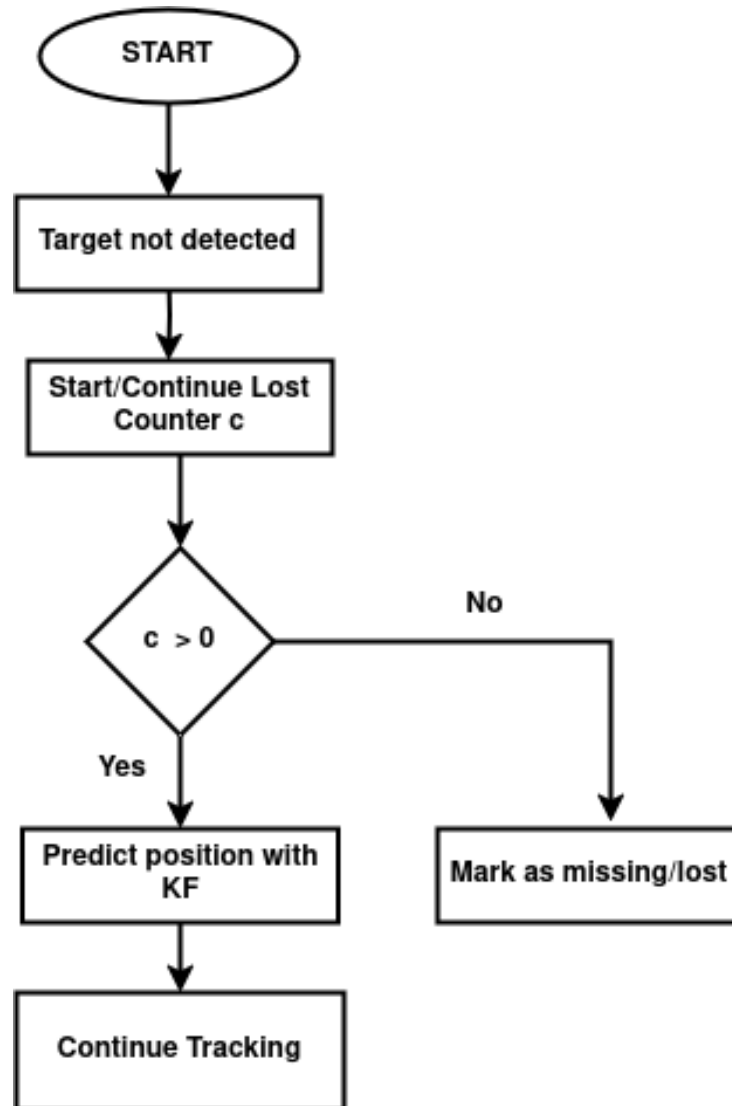


Figure 3.4: Flowchart representation of target missing mode

3. **Target Lost:** Declared after a set of consecutive frames where the target is absent, suggesting a potential loss or concealment. To prevent further deviation from the hidden target, the UAV will stop and hover, looking for potential redetection candidates until the target is redetected. In this stage, a relocation strategy could also be considered in future works to position the UAV in a better view to redetect it. The flowchart in Figure 3.5 shows the target lost mode handling.

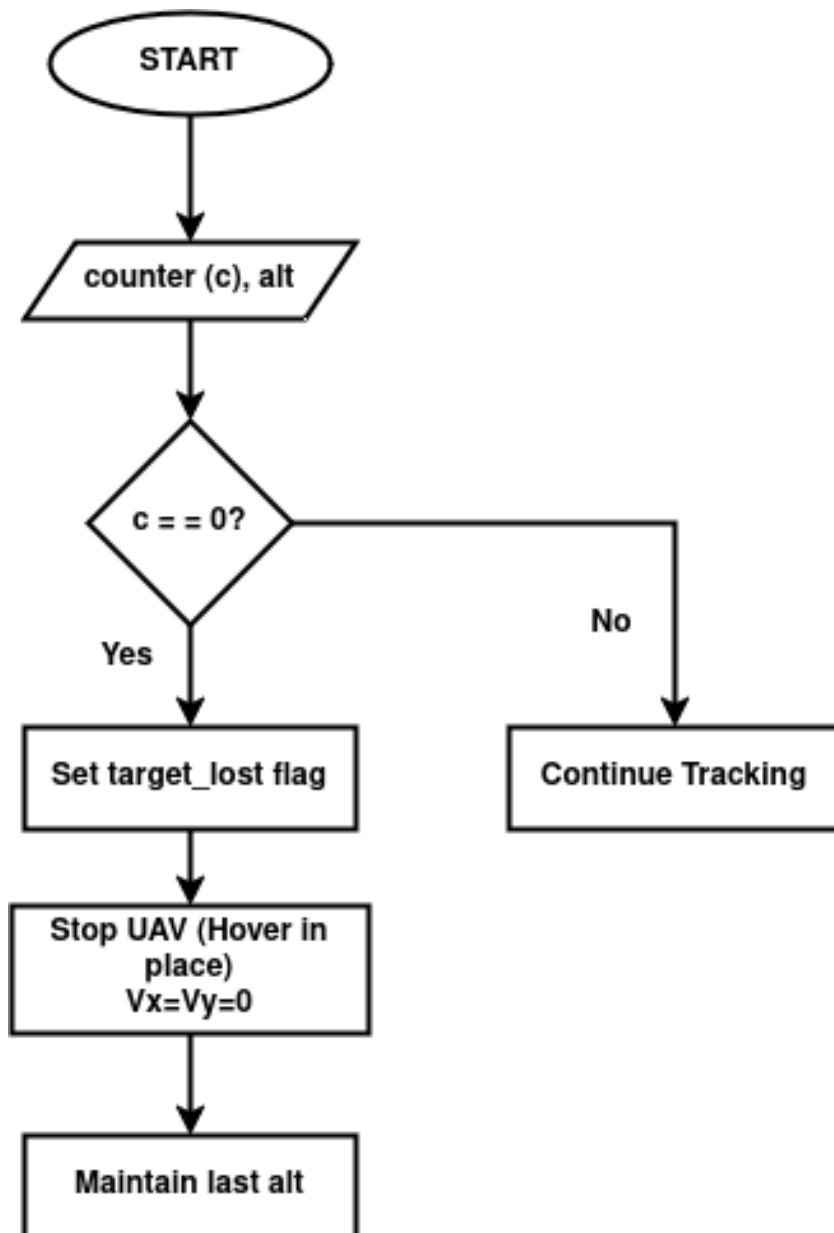


Figure 3.5: Flowchart representation of target lost mode

To perform the redetection process, the algorithm searches all detections for possible redetection candidates. Firstly, if the target ID is not found, it will attempt to check if a “Target ID Change” is in place. If no match is found, then it will enter the “Target Missing” mode and later the “Target Lost” mode. A viable candidate for a successful redetection must fulfill the following conditions:

1. It must represent a new detection not previously tracked, thereby exclud-

ing individuals already accounted for as they cannot be the target. This effectively excludes all the bystanders in the area and allows for redetections even in dynamic areas.

2. Candidate detections are evaluated based on a minimum interception threshold with the latest estimated position of the missing target, determined using an IoU approach (Equation 3.2). The bounding boxes are scaled to allow for a greater recovery range.
3. Among all the candidates that fulfill step 1 and step 2, it must be the one that scored the highest on step 2.

The IoU approach is a common evaluation method to determine similarity between two bounding boxes in an image. It is given by the relation in Equation 3.2 where B_A and B_B are the respective bounding boxes which can be visualized in Figure 3.6.

$$\text{IoU} = \frac{\text{Area}(B_A \cap B_B)}{\text{Area}(B_A \cup B_B)} \quad (3.2)$$

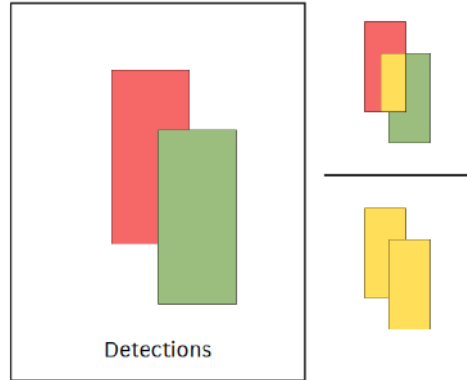


Figure 3.6: IOU representation

If the target is redetected during the “Target ID Change” or the “Target Missing” modes, the UAV will immediately resume regular operations and continue following

the target. If the redetection happened during the “Target Lost” state, then the system mode switcher will change the input to the controller accordingly, in order to resume target following operations after a complete stop. Further explanation will be given in the system mode switcher section.

3.2. Distance Estimation

Estimating distance is essential for effectively tracking a target. This estimation can be roughly determined using the detection bounding box from the “visual detection and tracking module” as shown in the Figure 3.7. In order to estimate distance d from the bounding box, we use a relation between the pixel height h of the target in the image and a tuned constant value C as shown in Equation 3.3. Moreover, UAV body frame is represented as F_b , camera frame as F_c , and world frame as F_w in the Figure 3.7.

$$d = \frac{C}{h} \quad (3.3)$$

The target’s height in the image is chosen as the reference instead of the bounding box area ($w * h$) [9] because it mainly depends on the actual height of the target and the distance to the target. Conversely, the width in the image can vary based on movement direction and limb position. The constant value can be pre-tuned for an average human height. For non-human targets like vehicles or robots, the bounding box area can be used.

To reduce susceptibility to observation noise, an exponential low-pass filter is applied in discrete time. The filtered estimate $d_{\text{low\_pass}}(k)$ at time step k is computed as a weighted combination of the current measurement $d(k)$ and the previous filtered estimate $d_{\text{low\_pass}}(k-1)$, as shown in Equation 3.4. The parameter $\alpha \in [0, 1]$ controls the smoothing factor, where higher values of α give more weight to recent measurements, allowing quicker response to changes, while lower values

emphasize historical estimates and reduce high-frequency noise.

$$d_{\text{low\_pass}}(k) = (1 - \alpha) \cdot d_{\text{low\_pass}}(k - 1) + \alpha \cdot d(k), \quad \alpha \in [0, 1] \quad (3.4)$$

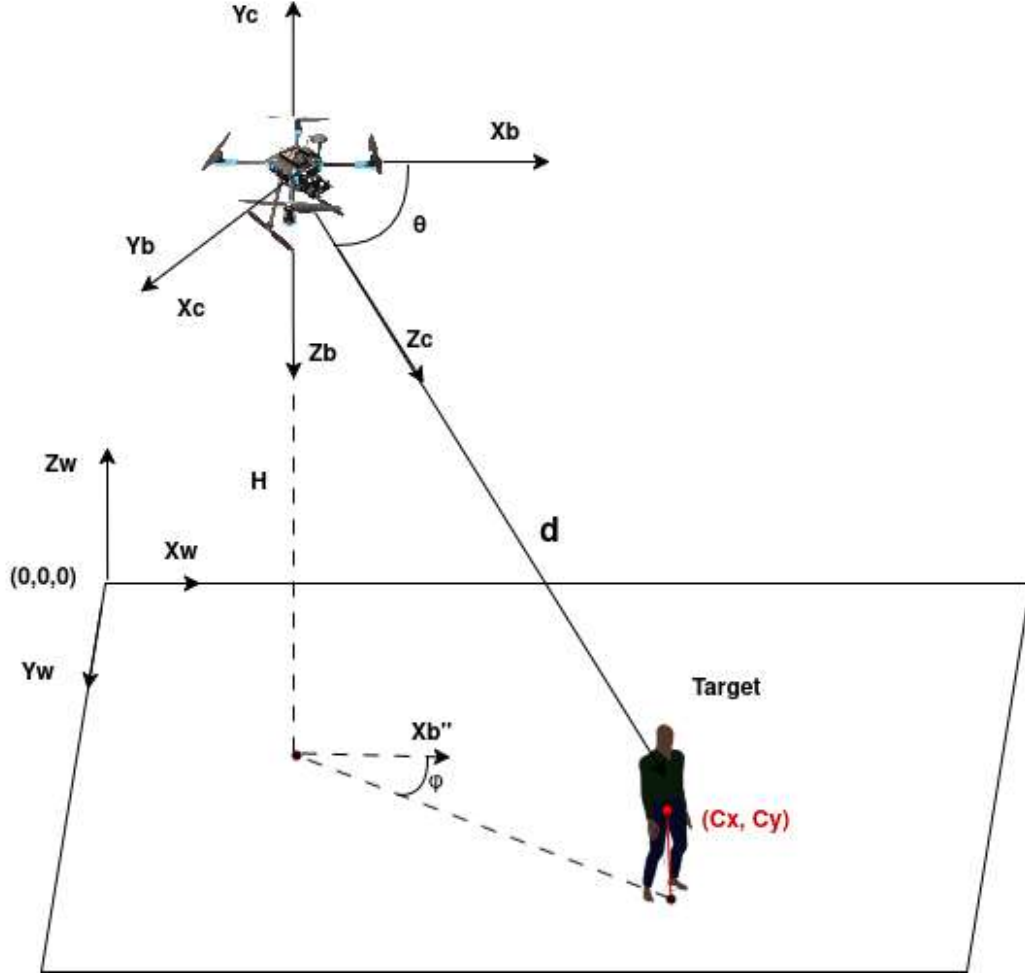


Figure 3.7: UAV-Target reference frames and relative distance d to the target

3.3. Flight Controller

The flight controller consists of three separate controllers to achieve precise 3D tracking of the target: one for yaw rate, one for altitude (vertical velocity), and one for the horizontal velocity of the UAV. This approach is inspired by [7, 8], using the deviation between the target's pixel location and the image frame's

centre as feedback to keep the UAV focused on the target. The yaw rate and altitude controllers are proportional (P) controllers aimed at centring the target in the image. To compute the control inputs, normalized references for the horizontal and vertical pixel positions are used, as shown in Equation 3.6. These normalized coordinates are based on the camera's intrinsic parameters, specifically the principal point coordinates ppx and ppy . The intrinsic matrix of the camera is defined as:

$$K = \begin{bmatrix} f_x & 0 & ppx \\ 0 & f_y & ppy \\ 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

Here, f_x and f_y are the focal lengths in pixels along the x and y axes, and (ppx, ppy) is the principal point.

Using these, the normalized image coordinates (X_N, Y_N) of the target, relative to the principal point, are computed as:

$$\begin{aligned} X_N &= \frac{C_x}{ppx} - 1 \\ Y_N &= \frac{C_y}{ppy} - 1 \end{aligned} \quad (3.6)$$

where C_x and C_y are the pixel coordinates of the target in the image frame.

The heading reference ϕ_{ref} for the controller and vertical velocity V_z are then computed using equations 3.7 and 3.8, respectively, where K_ϕ and K_H are the proportional gains:

$$\phi_{ref} = K_\phi \cdot X_N, \quad (3.7)$$

$$V_z = K_H \cdot Y_N. \quad (3.8)$$

The algorithm uses an aim-and-approach strategy, similar to the Proportional Navigation (PN) strategy in [5], to achieve target following. It uses the yaw rate

and altitude controllers to aim, and the horizontal velocity controller to approach the target. The control output for the velocity is calculated using a PI controller, taking the estimated distance as input. The horizontal velocity controller error is defined as:

$$error = d - d_{desired}. \quad (3.9)$$

where the desired distance $d_{desired}$ is defined by the user.

To achieve smooth control, the velocity signal V is passed through a slew rate limiter, which constrains the rate of change of the signal over time. This prevents sudden or aggressive maneuvers that may result in a loss of line of sight. The slew rate limiter also ensures the control output starts from zero, enabling gradual initial movement. The discrete-time rate of change of the velocity is given by:

$$\frac{V(t_k) - V(t_{k-1})}{t_k - t_{k-1}}$$

The slew rate limiter enforces the constraint:

$$\left| \frac{V(t_k) - V(t_{k-1})}{t_k - t_{k-1}} \right| \leq SR_{\max}$$

where $SR_{\max}$ is the user-defined maximum allowable slew rate.

In addition to the direct distance between the UAV and the target, the horizontal distance is considered a limiter to prevent the UAV from overshooting the target. This distance is calculated based on the current altitude of the UAV, an estimate of the target's height, and the assumption that the target is in the same plane as the measured altitude, as shown in Equation 3.10. This value can also be compromised by the quality of the altitude measurements, which is why it is only used as a limiter and not as the error feedback for the controller.

$$d_{\text{horizontal}} = \sqrt{d^2 - H^2}. \quad (3.10)$$

Figure 3.8 shows the workings of the high level controller. It takes the information from the RGB camera to compute the forward velocity, the vertical velocity and the yaw rate, which are then sent to the PX4 autopilot.

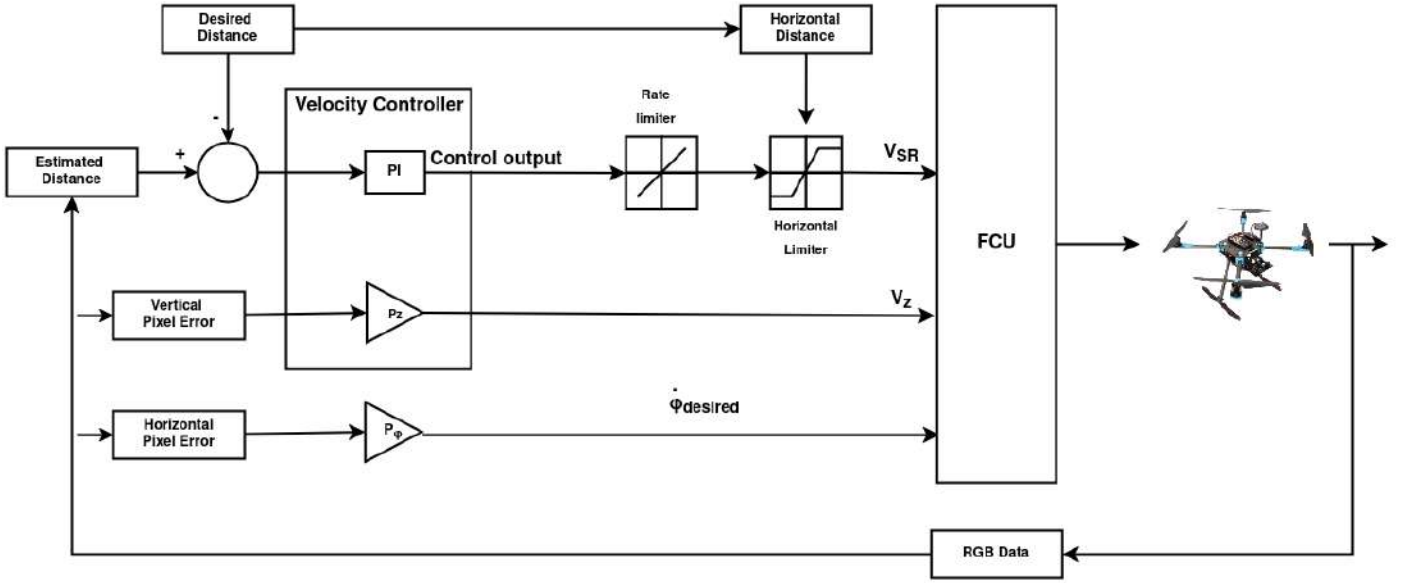


Figure 3.8: Simulink: UAV highlevel controller

3.4. System Mode Switcher

The system mode switcher is responsible for providing the low-level controller with appropriate control references based on the mode of the system. The system can operate in the following modes:

- **Search Mode:** No target has been detected and identified yet. The UAV performs a predetermined search pattern until the target is found.
- **Adjusting Mode:** Only the yaw rate and vertical velocity controllers are used to centre the target in the image.
- **Following Mode:** This may be defined as the mode for standard operations, when the target is identified and followed in 3D. This is also the chosen mode if a Target ID Change occurred.
- **Target Missing Mode:** As previously defined in Section 3.1.3, this mode activates when the target is not visible in the image.
- **Target Lost Mode:** Also defined in Section 3.1.3, if the target has been

missing for several consecutive frames it is assumed lost/hidden. Adequate procedures are taken to prevent further deviations from a target hidden behind an obstacle.

Figure 3.9 presents a flowchart representing the system modes and their respective interactions.

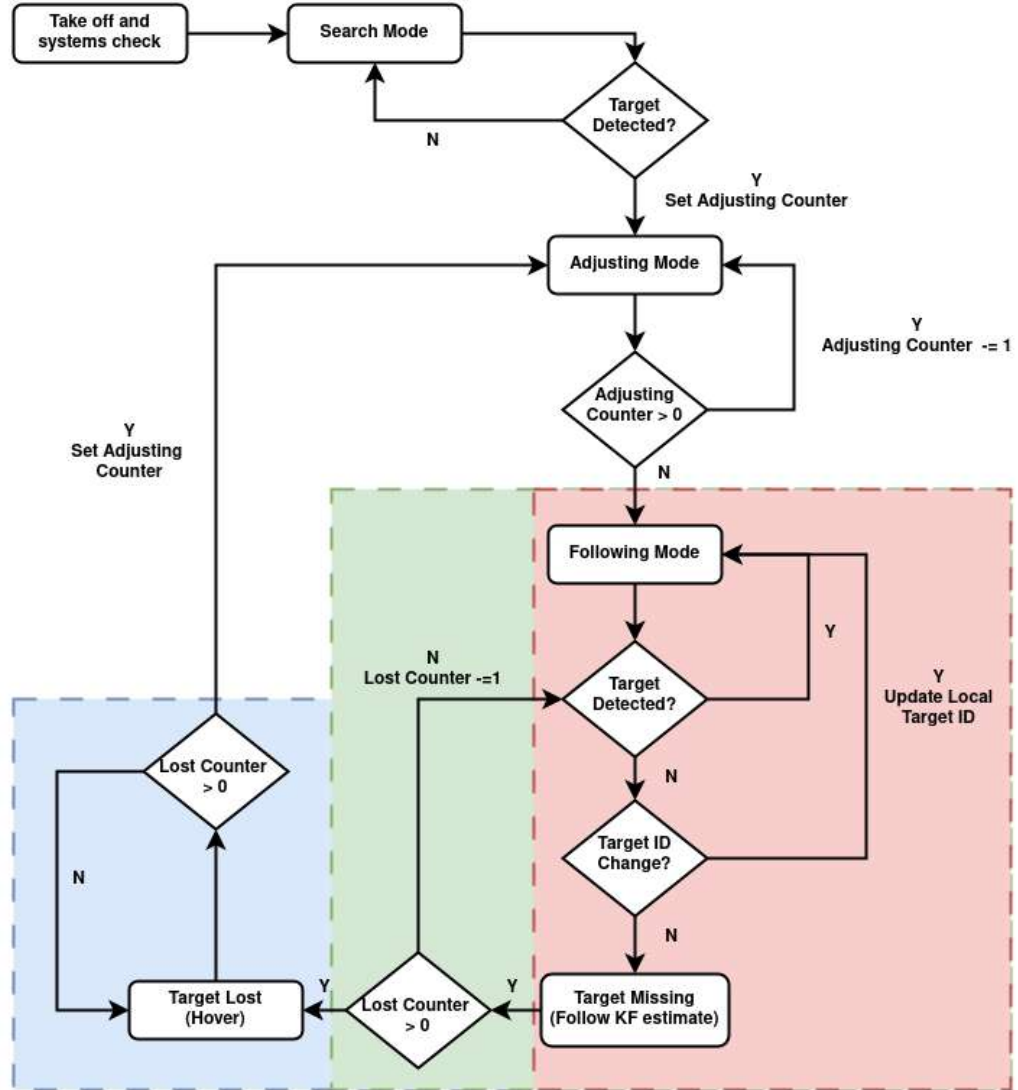


Figure 3.9: Diagram of system operation modes.

Following take-off and systems check, the UAV will enter “Search Mode”. In this mode, the UAV will perform a predetermined search pattern until the target is found. In this work, the UAV will perform a climb until the defined safety

altitude and slowly rotate until a person is detected. The first person detected will be set as the the target to follow.

The system allows the search mode to be redefined according to the specifications of the mission (e.g., making it possible for the user to select the specific target to follow). After the target is detected and identified, the UAV will enter “Adjusting Mode”, adjusting the target to the centre of the image as best as possible, considering altitude safety limits by controlling the altitude and yaw rate. This allows the system to have a more controlled first approach to the target. After the “Adjusting Mode” timer is over, the system enters regular operations with the “Following Mode”, which feeds the output from all the controllers to the autopilot.

Regarding the first recovery mode - “Target Missing Mode”—the system mode switcher will continue to send the commands from all the controllers however, the estimated distance and subsequently the error will be calculated using the predictions from the Kalman filter. Finally, in the second recovery mode - “Target Lost Mode” - the system mode switcher will set the UAV to hover. This is performed to prevent further deviation from the lost target, here assumed to be hidden behind some obstacle. The system will remain in this mode until a successful redetection is obtained. Unlike the “Target Missing Mode” that directly shifts to “Following Mode”, here, once the target is redetected, the system mode switcher will initiate the “Adjusting Mode” for a brief period and reset the rate limiter and the low pass filter previous values. This will ensure the target is properly redetected and followed.

4. Experiments and Results

4.1. Performance Metrics

The performance metrics are as follows:

- **UAV and Target Distances:** Total travel distance for UAV and target, respectively, in each experiment.
- **Visual Accuracy:** The percentage of frames where the target is correctly identified.
- **Estimation Error:** Average error between the estimated distance d and the real distance measured relative to ROS-Gazebo ground-truth; therefore, it is only used in simulation experiments.
- **Frames per Second (FPS):** Average rate of frames per second that are processed by the visual detection and tracking module, which in turn gives an accurate representation of the computational speed of the algorithm.
- **Number of ID Changes:** Measures target tracking performance, indicating how many times the ID assigned to the followed target changed, excluding corrections by redetection algorithms that adjust the target ID locally.
- **Controller Error:** Mean and standard deviation of the difference between the estimated distance and the desired reference distance, quantifying the controller's tracking precision.

4.2. Simulation Setup

The experiment was initially carried out in simulation for the validation of the algorithm. To assess system performance in open environments with multiple pedestrians and without airborne obstacles, multiple different tests were performed for the two trackers (i.e., BoT-SORT and ByteTrack) in a 900 square-meter obstacle-free area, featuring three pedestrians walking at an average speed of 1 to 1.5 m per second. Each test spans approximately 2 min, with randomized and not pre-set trajectories. The desired distance ($d_{desired}$) is set at 11 m. The choice of 11 meters as the desired distance was determined experimentally, balancing robustness and sensor limitations: shorter distances (6m–8m) led to occlusion and instability, while larger distances (12m–14m) caused oscillations and ID switches. The 10m–11m range proved optimal, minimizing target loss and maintaining consistent tracking in the simulated environment. In order to perform a comparison between MOT methods, the same world conditions (target movements) are replicated in the tests conducted using BoT-SORT and ByteTrack. The results are summarized in the quantitative results section 4.2.2.

4.2.1. Redetection Results

To comprehensively assess the redetection capabilities of the proposed system, several simulation scenarios were designed in which the target experienced important redetection modes: Target ID Change, target missing and target lost.

First, the effectiveness of the Target ID Change redetection mode was evaluated. As mentioned in Section 3.1.3, an ID change is a phenomenon in target tracking where the MOT assigns the target a different identity in consecutive frames. This is more common in dynamic environments and can be exaggerated due to sudden camera movements. Hence, the Target ID Change redetection mode is vital step to make the system able to track the target in dynamic environments by overwriting

the locally defined target ID with the newly assigned ID from the MOT algorithm, keeping accurate target tracking and following, despite MOT limitations. Figure 4.1 shows an example of several consecutive target ID Change redetections where the target is successfully redetected without interference in target following. These results demonstrate that the system is robust to ID changes, addressing one of the biggest limitations of MOT methods in these applications.

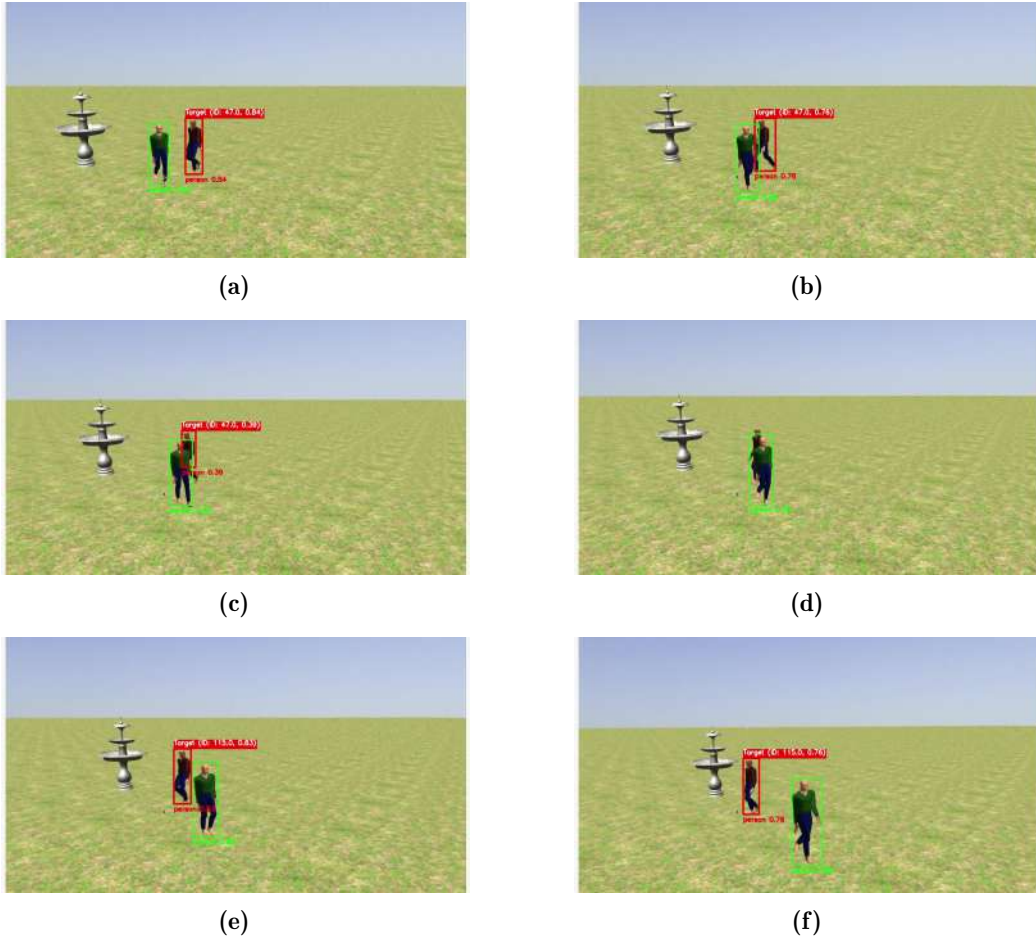


Figure 4.1: Target ID Change redetection experiment with another bystander: system recognizes and adapts to ID changes from the MOT module by updating the locally defined target ID. (a) to (c) Target followed with ID 47. (d) Occlusion due to bystander. (e) Target ID Change overwrite local target ID from 47 to 115. (f) Target followed with new ID

Followed by the ID change scenarios, the Target Missing Mode was assessed through scenarios with temporary occlusions, since this mode serves as an interme-

diary phase between the initial detection failure of the target and its categorization as lost. During this phase, an estimate of the position of the target is taken to navigate temporary occlusions without completely stopping the UAV. Figure 4.2 shows an example in which the target was obscured as it crossed behind bystanders. The results demonstrate a continuous target-following behavior even during the occlusion period by utilizing the Kalman-filter predictions during the target-missing mode accompanied with ID change.

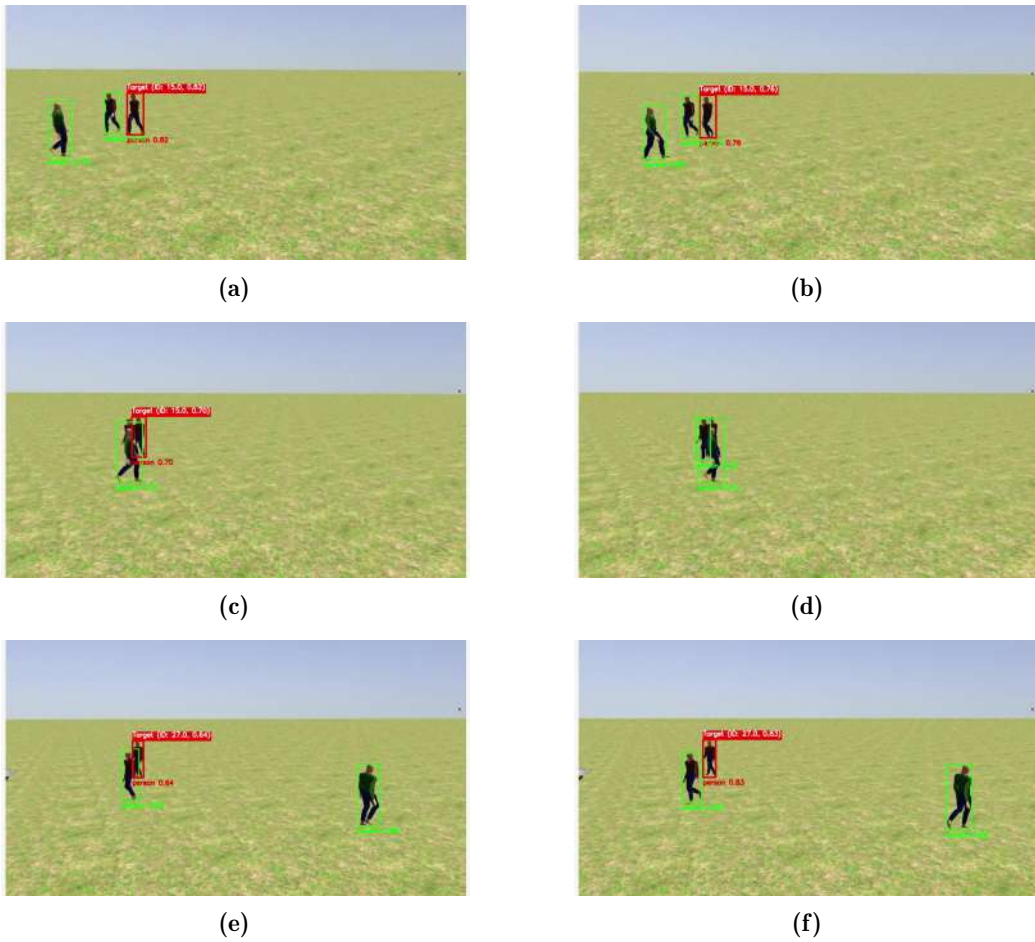
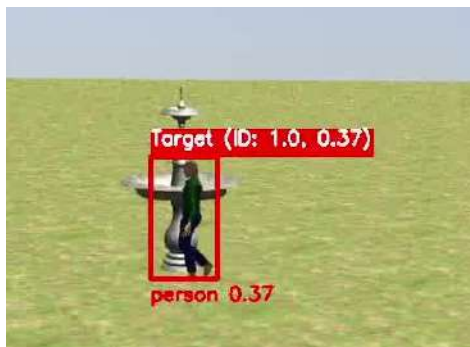


Figure 4.2: Target Missing redetection experiment: the target is redetected after temporary occlusion when crossing behind a bystander. (a) to (c) Following the target, Bystanders in the left. (d) Target crosses behind the bystander. Initiate Target Missing Mode. Target Missing, following Kalman filter estimate. (e) and (f) Target redetected and followed.

To test more complex redetection scenarios with full occlusions, a more chal-

lenging scenario was considered. That is, the target gets behind a non-person static obstacle in the terrain (fountain in the simulation). In this scenario, the capabilities of redetection are tested on a basic level without any interference from bystanders alongside the trajectory is planned with points which lead the target to go behind the fountain and pause there for a time period to achieve the target lost scenario, followed by turning back and walking straight (just for the experimentation). Figure 4.3 shows the target lost mode handling. The system tracks the target in the Following Mode for 4.3a. During Target Missing Mode (from Figure 4.3b until Figure 4.3c) the UAV attempts to follow the missing target by using the Kalman filter estimate. After some consecutive frames without redetecting the target, the system will switch to the Target Lost Mode where all movement is stopped (Figure 4.3c and 4.3d). Once the target is redetected (From Figure 4.3e) the system mode switcher will activate the Adjusting Mode to centre the target in the image which is then followed by the Following Mode (Figure 14f).



(a)



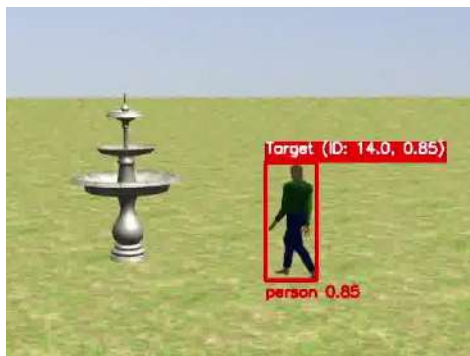
(b)



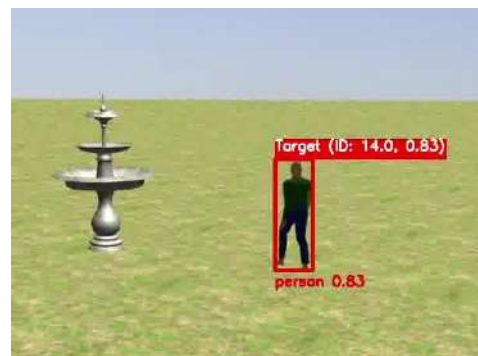
(c)



(d)



(e)



(f)

Figure 4.3: Target Lost Experiment—obstacle occlusion redetection scenario. Target successful redetection after full occlusion behind fountain in the simulation environment (a non-identifiable object). (a) Target Following Mode with ID 1. (b) Target hidden behind fountain. (c) and (d) Target lost. Stop movement. (d) Target reappears. (e) Target redetected with ID 14. Adjusting Mode. (f) Continue Target Following Mode.

4.2.2. Results and Analysis

Quantitative Results

The quantitative results based on the performance metrics explained in section 4.1, are summarized for the BoT-SORT tests in Table 4.1 and ByteTrack tests in Table 4.2. Across both experiments, the UAV successfully tracked and followed the target. For the BoT-SORT experiments, the UAV covered an average distance of approximately 102.45 m, while the target covered an average of 155.14 m. The average estimation error across the three trials was 1.33 m, with a standard deviation of ± 0.17 m, reflecting some variation due to the bounding box-based estimation approach.

In terms of computational performance, BoT-SORT achieved an average processing speed of $9.56 \text{ Hz} \pm 0.41 \text{ Hz}$, which is suitable for real-time UAV applications. Additionally, it maintained strong visual tracking accuracy with an average of 99.01%, and minimal ID changes (only 2 ID changes across all tests), indicating the algorithm's robustness in maintaining consistent target identification. Importantly, in the experiment 3 the first target detected was close to the camera field of view, followed by moving in the direction of UAV, despite this, BoT-SORT tracked the same target throughout whole experiment.

In contrast, ByteTrack, while showing slightly better runtime performance at $25.05 \text{ Hz} \pm 0.31 \text{ Hz}$, experienced a 31.25% increase in processing speed compared to BoT-SORT. However, this efficiency came at the cost of reduced tracking stability. Although the average visual accuracy remained high at 99.10%, there were 3 ID changes in just one test, suggesting occasional difficulties in maintaining consistent identity under dynamic conditions. ByteTrack also showed a slightly lower estimation error of $1.09 \text{ m} \pm 0.01 \text{ m}$, indicating a minor improvement in estimation precision.

In terms of controller performance, the mean controller error reveals that

ByteTrack offers slightly more stable control behavior. Across the three tests, ByteTrack maintained lower average errors compared to BoT-SORT’s more variable outcomes. This suggests that ByteTrack’s controller input remained more consistent, whereas BoT-SORT experienced occasional fluctuations. This is also because the simulation experiments for the ByteTrack tracker involving gains were tuned properly as compared to BoT-SORT setup. Hence more tuning of the gains for the BoT-SORT tracker would result in better performance.

Overall, while both algorithms demonstrated the ability to track and follow a target under varying conditions, BoT-SORT proved to be more reliable in maintaining consistent tracking identities, especially when faced with occlusions or dynamic movements. ByteTrack’s higher frame rate is beneficial for lightweight applications, but its lack of motion compensation and increased ID switching makes BoT-SORT a more robust choice for real-world UAV-based target following scenarios.

Table 4.1: BoT-SORT data from randomized tests.

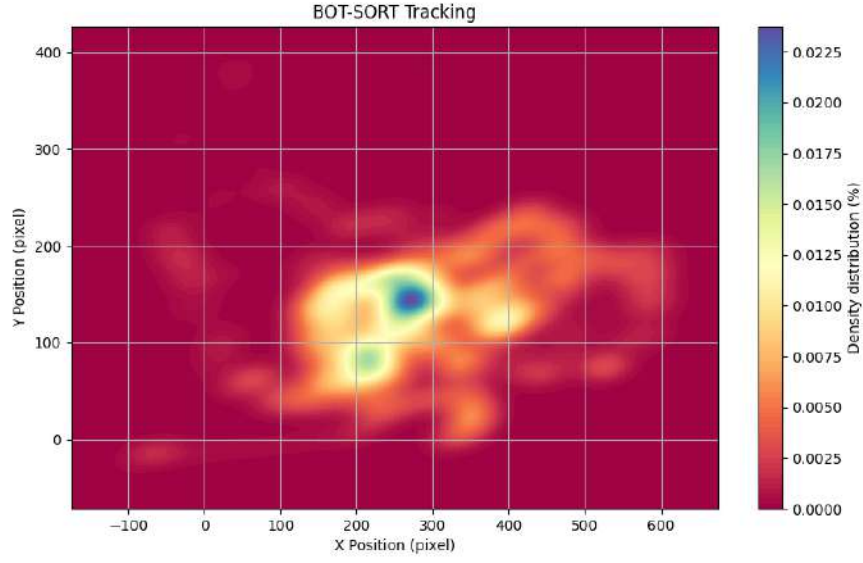
Test No.	Time (s)	Distance UAV (m)	Distance Target (m)	Visual Acc. (%)	Estimation Error (m)	FPS (Hz)	No. ID Change	Mean Con- troller Error (m)	
1	155.92	80.75	122.94	99.70	1.13	9.03	0	0.34	±
								1.94	
2	209.81	121.84	177.43	97.89	1.38	10.04	2	0.93	±
								4.08	
3	183.00	104.75	165.04	99.43	1.47	9.62	0	0.03	±
								2.71	

Table 4.2: ByteTrack data from randomized tests.

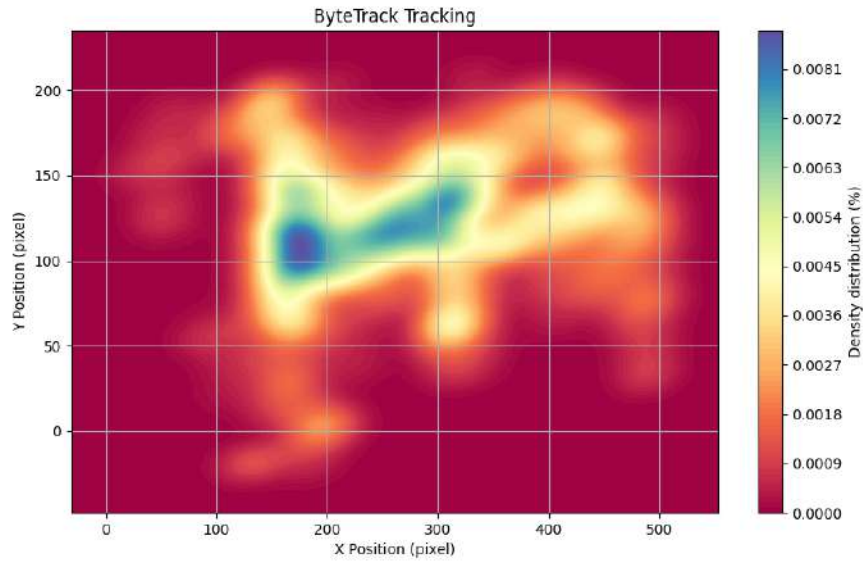
Test No.	Time (s)	Distance UAV (m)	Distance Target (m)	Visual Acc. (%)	Estimation Error (m)	FPS (Hz)	No. ID Change	Mean Con- troller Error (m)
1	144.15	90.72	129.03	97.75	1.10	25.39	3	0.49 ± 3.44
2	220.94	105.52	202.73	99.80	1.07	25.14	0	0.22 ± 1.79
3	169.70	82.34	153.50	99.75	1.09	24.63	0	0.31 ± 1.80

Qualitative Results

To evaluate the system’s ability to maintain line of sight and keep the target centered in the image, heatmaps of the central position of the target were compiled for each tracker during a single experiment, as shown in Figure 4.4. The results are generally similar for both trackers, with the target predominantly remaining near the center of the frame. However, the ByteTrack results exhibit some off-centered or dispersed regions. Vertical deviations in the target position are likely due to the absence of gimbal stabilization, which causes forward and backward movements of the platform to induce pitch motions in the camera. Introducing camera stabilization is expected to significantly improve performance by decoupling translational motion from camera orientation.



(a) BoT-SORT tracking



(b) ByteTrack tracking

Figure 4.4: Spatial heatmap showing the trajectory of the mobile target's centre within the image frame during single experiment. This visualization highlights how the target was consistently tracked and kept within view during the target-following process, showing the areas of highest occurrence over time.

Figure 4.5 shows the real-time trajectories of a UAV (blue) and a moving target (orange) in a 2D plane i.e. X and Y from one of the short experiments

performed, with the UAV starting near (0,0) and the target near (-10,-5) with the respective markers indicating the same. The UAV's path demonstrates successful vision-based tracking, as it adjusts its trajectory to follow the target's motion.

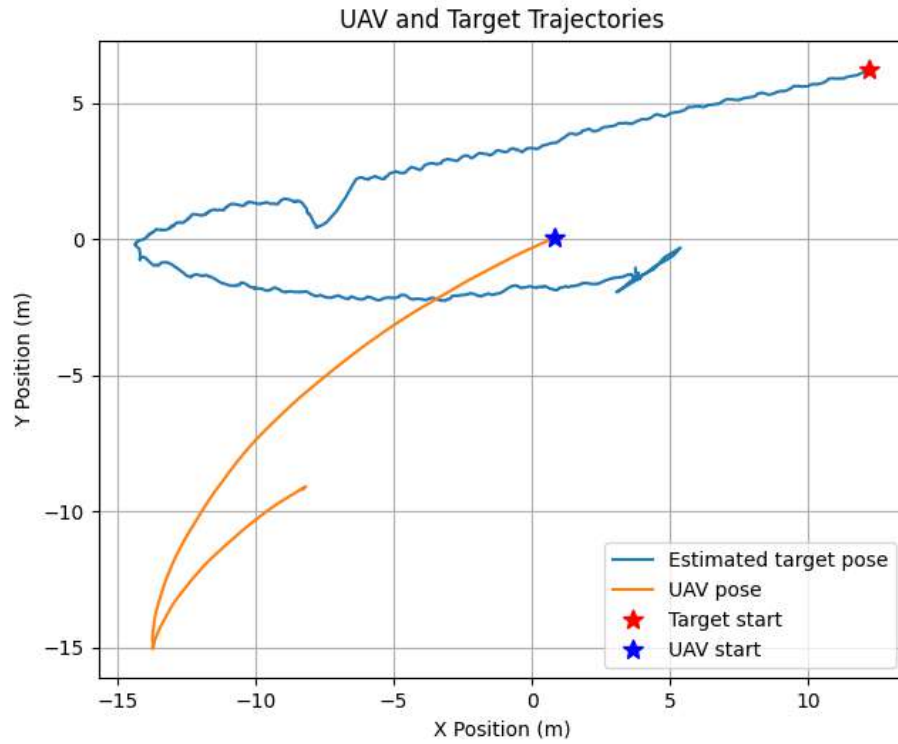
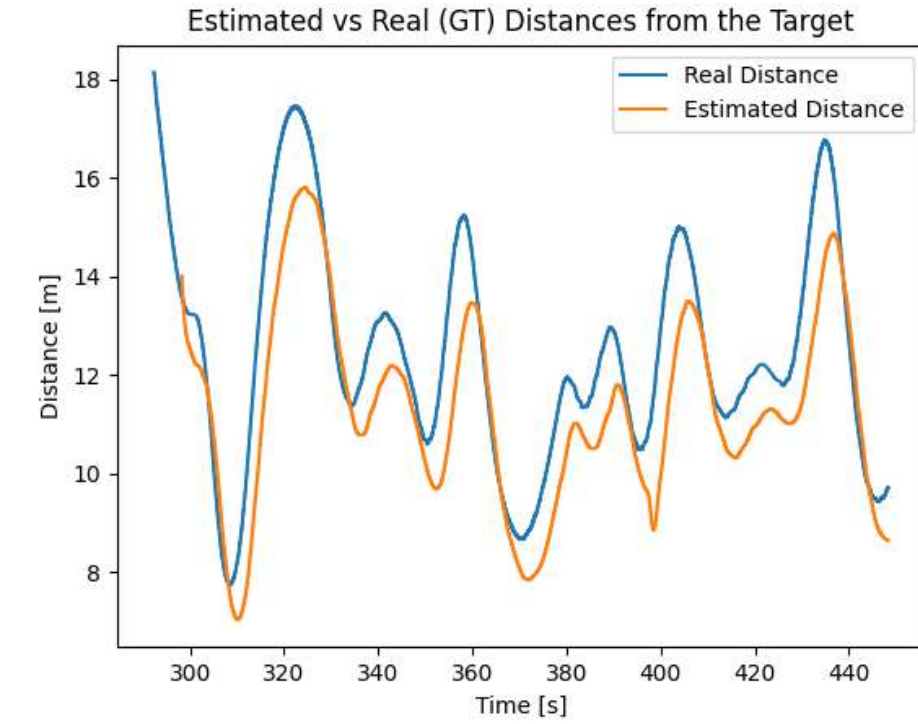
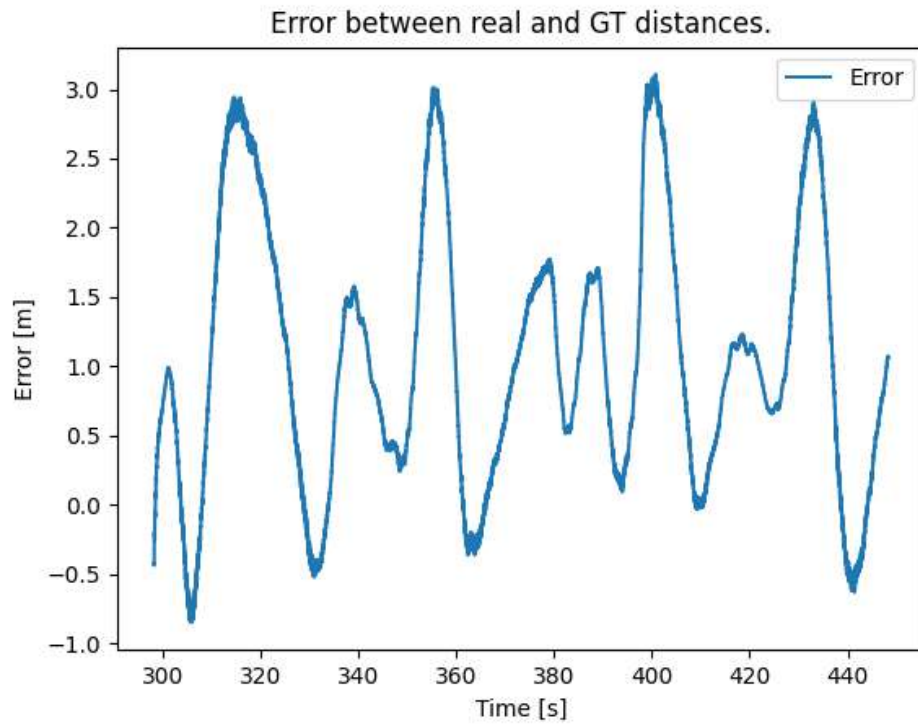


Figure 4.5: Drone and Target 2D (X-Y) Trajectories

Figure 4.6 presents the estimated versus actual distances, along with the respective distance estimation error, from one of the BoT-SORT experiment. A demonstration video of the full experiment using YOLOv8 and BoT-SORT can be viewed at: (<https://youtu.be/7RVzWSg1Pec>).



(a)



(b)

Figure 4.6: Estimated distance and respective error (distance estimation error) over time for one of the BoT-SORT experiment.

The Controller Tracking Error plot in Figure 4.7 shows the difference between the UAV's estimated distance to the target and the desired distance (11 m), with positive values indicating the UAV is too far and negative values indicating it is too close. This experiment was performed with ByteTrack tracker where The mean error (0.19 m) and standard deviation (1.82 m) demonstrate stable tracking performance, with oscillations reflecting dynamic adjustments. However, more work on tuning the system is required including the gains used for the controller for both ByteTrack and BoT-SORT trackers.

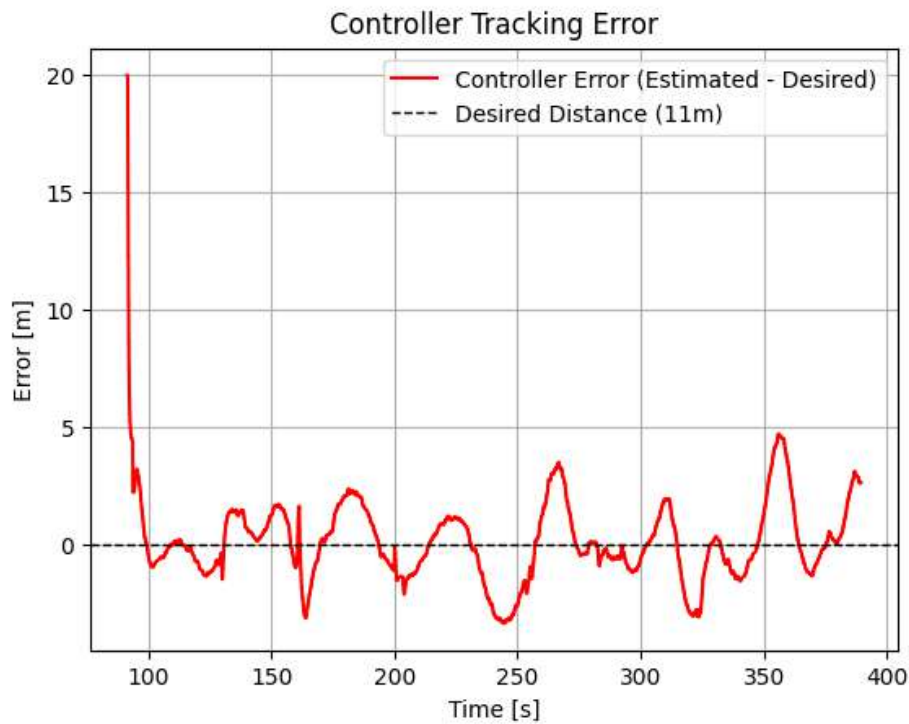


Figure 4.7: Tracking error of the drone controller

5. Conclusion

This work introduces a vision-based UAV target tracking system that uses multi-target data to improve re-identification in complex dynamic environments where partial or full occlusions are possible. It demonstrates strong performance in simulation tests, maintaining target focus even during occlusions and abrupt movements or direction changes. The 3D controller ensures stable target following by keeping the target centered in the image, and a comparison of tracking algorithms highlights superior precision of BoT-SORT especially in reducing ID switches despite the advantage in speed of ByteTrack.

5.1. Future Work

Despite the progress made in vision-based tracking and following using a UAV, further research and development opportunities remain:

- **Handling Fast Target Motion:** The system currently faces limitations related to maximum target acceleration and sudden direction changes, which can result in loss of line of sight (LOS) if the target moves rapidly toward the UAV. To mitigate this, future work could explore predictive motion modeling using recurrent neural networks (RNNs) or transformer-based trackers that learn temporal patterns of movement and anticipate future positions of the target [24]. Additionally, increasing the frame rate of detection and reducing controller latency could enhance the responsiveness of the UAV.

- **Camera Gimbal Stabilization:** Furthermore, maintaining precise target centering is hindered by the absence of camera stabilization, since the pitch and roll of the UAV directly influence the camera's orientation. This coupling affects the target's position in the vertical axis during forward or backward motion and on the horizontal axis when the UAV rolls. Future improvements could involve incorporating a gimbal to decouple camera motion from UAV dynamics, along with implementing path planning algorithms to optimize flight paths and avoid obstacles during application deployment.
- **Enhancing Performance in Low-Light with Multi-modal (RGB-T) Fusion:** To enable functionality in low-light conditions, the integration of RGB and thermal data through multimodal sensor fusion (RGB-T) is also recommended. Moreover, learning-based approaches to infer depth information from RGB data should be considered. Estimating depth directly from RGB images using deep learning models could significantly enhance the UAV's ability to estimate and maintain distance to the target, building upon the current tracking system's capabilities.
- **Real World Deployment:** Due to time constraints within the research group and undesirable weather conditions, real-world tests could not be performed, however, the hardware setup was exhaustively prepared for it including setting up the Raspberry Pi5 with camera module 3 along with HAILO AI HAT+ to increase processing required for detection and tracking module. Including such tests in future work would provide valuable information on the performance of the system under practical conditions.

In conclusion, this system provides a solid foundation for future research by addressing the challenge of ID changes and presenting a real-world application of multi-object tracking (MOT) in UAVs.

6. References

- [1] U. O. Matthew, J. S. Kazaure, A. Onyebuchi, O. O. Daniel, I. H. Muhammed, and N. U. Okafor. Artificial Intelligence Autonomous Unmanned Aerial Vehicle (UAV) System for Remote Sensing in Security Surveillance. In *2020 IEEE 2nd International Conference on Cyberspace (CYBER NIGERIA)*, pages 1–10, Abuja, Nigeria, 2021. doi: 10.1109/CYBERNIGERIA51635.2021.9428862.
- [2] H. Ali, L. Y. Hang, T. Y. Suan, V. R. Polaiah, M. I. F. Aluwi, A. A. Mohd Zabidi, and M. Elshaikh. Development of surveillance drone based internet of things (IoT) for industrial security applications. *Journal of Physics: Conference Series*, 2107:012018, 2021. doi: 10.1088/1742-6596/2107/1/012018. International Conference on Man Machine Systems (ICoMMS 2021), 19–20 October 2021, Perlis, Malaysia.
- [3] Fredrik Heintz, Piotr Rudol, and Patrick Doherty. From Images to Traffic behavior - A UAV Tracking and Monitoring Application. In *2007 10th International Conference on Information Fusion*, pages 1–8, 2007. doi: 10.1109/ICIF.2007.4408103.
- [4] J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han. Search and Rescue Rotary-Wing UAV and Its Application to the Lushan Ms 7.0 Earthquake. *Journal of Field Robotics*, 33(3):290–321, 2016.
- [5] X. Liu, Y. Yang, C. Ma, J. Li, and S. Zhang. Real-Time Visual Tracking of

- Moving Targets Using a Low-Cost Unmanned Aerial Vehicle with a 3-Axis Stabilized Gimbal System. *Applied Sciences*, 10(15):5064, 2020.
- [6] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu. An Autonomous Vision-Based Target Tracking System for Rotorcraft Unmanned Aerial Vehicles. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1732–1738, Vancouver, BC, Canada, September 2017.
- [7] Y. Feng, D. Wang, and K. Yang. Research on Target Tracking Algorithm of Micro-UAV Based on Monocular Vision. *Journal of Robotics*, 2023:6657120, 2023.
- [8] D. Luo, P. Shao, H. Xu, and L. Wang. Autonomous Following Algorithm for UAV Based on Multi-Scale KCF and KF. In *Proceedings of the 2023 4th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 430–436, Nanjing, China, June 2023.
- [9] H. Wei. A UAV Target Prediction and Tracking Method Based on KCF and Kalman Filter Hybrid Algorithm. In *Proceedings of the 2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pages 711–718, Guangzhou, China, January 2022.
- [10] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.
- [11] S. Yadav and S. Payandeh. Critical Overview of Visual Tracking with Kernel Correlation Filter. *Technologies*, 9(4):93, 2021.
- [12] S. Liu, X. Li, H. Lu, and Y. He. Multi-Object Tracking Meets Moving UAV. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and*

- Pattern Recognition (CVPR)*, pages 8866–8875, New Orleans, LA, USA, June 2022. IEEE.
- [13] N. Aharon, R. Orfaig, and B.Z. Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *arXiv preprint arXiv:2206.14651*, 2022.
 - [14] M. Hussain. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines*, 11(6):677, 2023.
 - [15] T. Li, Z. Li, Y. Mu, and J. Su. Pedestrian Multi-Object Tracking Based on YOLOv7 and BoT-SORT. In L. Shen and G. Zhong, editors, *Proceedings of the Third International Conference on Computer Vision and Pattern Analysis (ICCPA 2023)*, volume 12754, page 127541I, Hangzhou, China, 2023. SPIE.
 - [16] S. Yan, Y. Fu, W. Zhang, W. Yang, R. Yu, and F. Zhang. Multi-Target Instance Segmentation and Tracking Using YOLOV8 and BoT-SORT for Video SAR. In *Proceedings of the 2023 5th International Conference on Electronic Engineering and Informatics (EEI)*, pages 506–510, Wuhan, China, 2023.
 - [17] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO. <https://github.com/ultralytics/ultralytics>, 2023. Accessed on 1 February 2024.
 - [18] C.Y. Wang, I.H. Yeh, and H.Y.M. Liao. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv preprint arXiv:2402.13616*, 2024.
 - [19] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. YOLOv10: Real-Time End-to-End Object Detection. *arXiv preprint arXiv:2405.14458*, 2024.

- [20] Lh. He, Yz. Zhou, L. Liu, et al. Research on Object Detection and Recognition in Remote Sensing Images Based on YOLOv11. *Scientific Reports*, 15:14032, 2025. doi: 10.1038/s41598-025-96314-x. URL <https://doi.org/10.1038/s41598-025-96314-x>.
- [21] D. Ferreira and M. Basiri. Dynamic Target Tracking and Following with UAVs Using Multi-Target Information: Leveraging YOLOv8 and MOT Algorithms. *Drones*, 8(9):488, 2024. doi: 10.3390/drones8090488. URL <https://doi.org/10.3390/drones8090488>.
- [22] N. L. Baisa. Derivation of a Constant Velocity Motion Model for Visual Tracking. *arXiv preprint arXiv:2005.00844*, 2020.
- [23] Roger R. Labbe. *FilterPy: Python Library for Kalman and Bayesian Filters*, 2016. URL <https://github.com/rlabbe/filterpy>. Version 1.4.4.
- [24] Alexandre Alahi, Kratarth Goel, Varun Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/Alahi_Social_LSTM_Human_CVPR_2016_paper.html.

16

**Globally Consistent Mapping in Indoor and Outdoor Environments
Using Hybrid LiDAR SLAM
Zewdie Habtie Sisay**



EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

ERASMUS MUNDUS JOINT MASTER IN
INTELLIGENT FIELD ROBOTIC SYSTEMS

Globally Consistent Mapping in Indoor and Outdoor Environments Using Hybrid LiDAR SLAM

Author:

Zewdie Habtie Sisay

MSc Intelligent Field Robotic Systems (IFRoS)

Internal supervisor:

Zoltán Istenes

Professor

External supervisor:

Mohammad Albaja

Associate Professor, SMART

Budapest, 2025

Acknowledgements

First and foremost, I would like to thank the Almighty God for His abundant grace, guidance, and strength throughout my academic journey. Without His blessings, this achievement would not have been possible.

I would also like to express my sincere gratitude to the IFROS program and its organizers, as well as the professors, for providing me with the opportunity to pursue this master's degree and for the invaluable knowledge I have gained throughout the program.

I extend my heartfelt thanks to my industrial supervisor, Mohammad Aldibaja, and my university supervisor, Zoltán Istenes, for their continuous guidance and constructive feedback during the course of my thesis. I am also sincerely grateful to Professor Abeje Mersha, leader of the Smart Mechatronics and Robotics Research Lab, for offering me the opportunity to carry out my thesis work in his lab and for the supportive research environment he provided.

I would also like to thank the Smart Mechatronics and Robotics Research Group and Saxion University of Applied Sciences for their support and for fostering a collaborative and enriching environment during my thesis work.

I am especially thankful to my beloved wife, Genet, and my children, Nathan and Hemen, who, despite being far away in Ethiopia, have been a constant source of strength and encouragement throughout my studies. Their love, patience, and unwavering support gave me the courage to persevere and have been the deepest inspiration behind this journey.

Last but not least, I would like to express my gratitude to my friends, whose encouragement, companionship, and moral support have greatly contributed to making this journey more fulfilling and enjoyable. Their kindness and understanding have been invaluable throughout my studies and thesis work.

Abstract

Simultaneous Localization and Mapping (SLAM) is a foundational component of autonomous robotic systems, enabling them to navigate and interact with unknown environments. While LiDAR-based SLAM methods offer high-precision geometric mapping, they often suffer from accumulated drift over long trajectories and lack global consistency, particularly in environments with loop closures. This thesis presents a hybrid LiDAR SLAM framework that integrates LiDAR-Inertial Odometry (LIO), loop closure detection, Pose Graph Optimization (PGO), and Hierarchical Bundle Adjustment (HBA) to generate globally consistent maps in both indoor and outdoor settings.

The proposed system employs FAST-LIO2 for real-time state estimation, Scan Context for loop closure detection, and PGO to correct drift. To refine global map consistency, HBA is applied as an offline optimization step, minimizing misalignment in revisited areas. The framework is evaluated on both public datasets (KITTI, MulRan) and custom datasets collected using a mobile robot equipped with an Ouster OS1-128 LiDAR and IMU. Quantitative evaluation using trajectory accuracy, map consistency metrics, and runtime analysis demonstrates that the system achieves improved global consistency over baseline methods.

The results confirm that combining LIO with graph-based global optimization and HBA refinement can significantly enhance SLAM performance in large-scale and loop-rich environments. This hybrid approach provides a scalable and robust solution for long-term autonomy in dynamic and diverse environments.

Contents

Acknowledgements	i
Abstract	ii
List of Figures	iii
List of Tables	v
List of Algorithms	vi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Research Questions	2
1.4 Thesis Structure	2
2 Literature Review	4
2.1 Introduction	4
2.2 Probabilistic notations	5
2.3 LiDAR Odometry and LiDAR Inertial Odometry	6
2.4 Lie Theory and 3D pose representation	7
2.4.1 Lie groups	8
2.4.2 Lie algebra	9
2.5 Introduction to Bundle Adjustment	9
2.5.1 LiDAR Bundle Adjustment	10
2.6 Overview and types of SLAM	12
2.6.1 Online SLAM	12
2.6.2 Full SLAM	13
2.7 Related works	15

3	Methodology	18
3.1	System Overview	18
3.2	Description of FAST-LIO2	19
3.2.1	Prediction Step	20
3.2.2	Update Step	20
3.3	Loop Closure Detection	21
3.3.1	Scan Context	22
3.4	ICP for computing loop closure constraint	25
3.5	Optimization with Pose Graph	26
3.6	Global refinement using Bundle Adjustment	27
3.6.1	Bottom-up Process	28
3.6.2	Top-Down Process	29
4	Experiments and Results	30
4.1	Experimental Setup	30
4.1.1	Hardware and Platform	30
4.1.2	Dataset	31
4.2	Evaluation metrics	32
4.3	Results	33
4.3.1	Trajectory accuracy	33
4.3.2	Map quality and consistency	35
4.3.3	Runtime performance	38
4.4	Comparison with state of the art	40
5	Discussion	43
5.1	Interpreting the results	43
5.1.1	Trajectory and Map Accuracy	43
5.1.2	Discussing the ATE and MME comparison results	44
5.2	Challenges	45
6	Conclusion	47
6.1	Summary of Findings	47
6.2	Research questions revisited	48
6.3	Contributions	48
6.4	Future work	49

A Appendices	50
A.1 Trajectories of public datasets	51
A.2 Results for MulRan KAIST3 sequence	52
A.3 Results for KITTI sequence 00	53
A.4 Results for Saxion11(Indoor)	54
A.5 Results for Saxion02	55
 Bibliography	 55
 Acronyms	 61

List of Figures

2.1	Relationship between Lie Groups and Lie algebras[15]	8
2.2	Visual BA	10
2.3	A plane in space and corresponding feature points drawn from different scans	11
2.4	Probabilistic Bayesian net SLAM model	14
2.5	Full SLAM model with factor graph	14
3.1	Block diagram of the proposed system pipeline.	18
3.2	Loop detected with Scan Context. a) current scan(source), b) target scan in the map, c) before alignment d) after alignment with ICP	22
3.3	A 2D bin encoding of the 3D point cloud in a single scan[29]	23
3.4	A point cloud from a single 3D scan is encoded into a Scan Context. An N_r dimensional ring key is then extracted and used to build a KD-tree for retrieving nearest candidates. These candidates are compared with the query, and the closest one that meets the acceptance threshold is selected as a loop closure[29].	24
3.5	Error between actual measurement $\mathbf{z}_{ij}$ and expected measurement $\hat{\mathbf{z}}_{ij}$. . .	26
3.6	Pose graph for KITTI00 sequence before optimization, nodes are shown in red dots, and factors are shown in green lines. The blue trajectory shows the path after the pose graph is optimized.	27
3.7	Pyramid structure of the HBA with layer number $l=3$, stride size $s=3$ and window size $w=6$ [20].	27
3.8	HBA optimization: Bottom-Up and Top-Down Process[20]	29
4.1	a) Ouster OS1-128 LiDAR with built-in IMU. b) Sensor coordinate frame [40]	30
4.2	APE of FAST-LIO2 and PGO trajectories for Saxion-01 dataset w.r.t HBA trajectory.	33
4.3	Axis-wise comparison of the trajectories of FAST-LIO2, PGO, and HBA .	34

4.4	Comparison of trajecories for Saxion-01 dataset.	35
4.5	Map generated by each stage of the mapping pipeline for saxion01. (a) FAST-LIO2 top view. (b) PGO top view. (c) FAST-LIO side view. (d) PGO side view. (e) HBA	37
4.6	Mapping result for saxion03	38
4.7	Processing times of PGO module	39
4.8	Trajectory Error comparison with other SLAM methods.	41
4.9	Greenhouse map	42
5.1	Loop detection failure case in lane-level later shift and reverse revisit . . .	45
5.2		46
5.3	Effect of loop detection failure in the map	46
A.2	Map generated by each stage of the mapping pipeline for MulRan KAIST dataset sequence 03. (a) Map generated by FAST-LIO2, (b) Map after optimizing with PGO, (c) and (d) show segments from (a) and (b) zoomed in respectively.	52
A.3	(a) The zoomed in segment of map in Figure A.2d, (b) After refinement with HBA	52
A.4	Map visualization for KITTI00 sequence. (a) BEV of map generated by FAST-LIO2. (b) BEV of map after PGO optimization. (c) BEV of FAST-LIO2 and PGO trajectories. (d)Side view of FAST-LIO2 map. (e) Side view the optimized map. (f) Side view of FAST-LIO2 and PGO trajectories	53
A.5	Visualization of trajectories for KITTI00. a) BEV trajectories . b)Axis-wise comparison of the trajectories.	53
A.6	Mapping result for saxion11 [Indoor]	54
A.7	Mapping result for Saxion02 after refinement with HBA	55

List of Tables

3.1	Parameters for ICP (Point-to-Point)	25
4.1	Dataset used for evaluating performance of our mapping module	32
4.2	Extrinsic calibration between IMU and LiDAR for the different datasets we used	32
4.3	Absolute Pose Error (APE) with respect to translation (in meters) for saxion-01 .	33
4.4	RMSE of the APE for each Saxion dataset before and after PGO	34
4.5	MME(μ, σ) of map generated from the different datasets after each mapping stage. The MME values in bold demonstrate the best quality of the map and those in <u>underline</u> show the second best quality of the map.	38
4.6	MME comparison with state of the art.	40
4.7	Comparison of RMSE of the APE(translation) for the public dataset using our method and other SLAM methods. The values for the method that resulted in smallest error is shown in bold.	40

List of Algorithms

1	LiDAR-Inertial Odometry with Global Optimization	19
2	Loop Detection with Scan Context	24

Chapter 1

Introduction

1.1 Motivation

In recent years, robotics has evolved with the goal to develop intelligent robots that can assist or replace human beings in performing tasks. In order for these robots to operate effectively and efficiently in diverse environment, a set of modules that define the robot autonomy such as path planning, localization, mapping, perception, and manipulation are required. These modules are interconnected to each other in a sense that one can be an input to the other. In similar manner to how we humans understand and interact with our environment, robots also need to have these modules tightly integrated.

Map is an important prerequisite that helps a robot determine the pose of objects within its environment, enabling it to operate and perform tasks such as navigation, obstacle avoidance, and path planning. Both mapping and localization modules are usually combined and referred to as SLAM module where the robot builds the map while continuously localizing itself inside it. The planning module allows the robot to plan a path from its current pose (position, and orientation) to a goal pose. An algorithm that assists the robot to follow the planned path, a controller, is also considerably essential entity. Using perception module, robot senses and interpret the environment. In same way we humans use our sensing organs to sense our surrounding, robot uses its onboard sensors such as camera, Light Detection and Ranging (LiDAR), Radio Detection and Ranging Radio Detection and Ranging (RADAR), Inertial Measurement Unit (IMU), Global Positioning System (GPS) and other sensors. The manipulation module enables the robot to interact with objects in its environment by performing tasks such as grasping,

picking, placing, and etc.

Map being an important prerequisite and SLAM being the state of the art solution to generating it, there are several techniques to solving it in the literature using different or similar methods. The two main techniques are online and full SLAM. These techniques rely on different sensor modalities with LiDAR and Visual SLAM being two the most widely used approaches.

1.2 Objectives

The primary objectives of the thesis are:

- To develop robust LiDAR SLAM pipeline that maintains global consistency of map both for indoor and outdoor environments by integrating LiDAR Inertial odometry, graph SLAM, LiDAR bundle adjustment.
- To incorporate robust loop closure detection technique with outlier rejection.
- To evaluate the systems performance against state-of-the-art SLAM methods using multiple datasets, analyzing metrics such as trajectory accuracy, map consistency, computational efficiency.

1.3 Research Questions

- Is LiDAR inertial odometry sufficient to generate accurate map of complex environments?
- What are the limitations of Graph SLAM in generating consistent map especially in revisited locations?
- How effective is it to integrate LIO, Graph SLAM, and LiDAR Bundle Adjustment for generating globally consistent map of environments?

1.4 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 discusses the literature review focusing in background foundations and describing the most related topics; LIO, types

of SLAM, and LiDAR Bundle Adjustment. Chapter 3 provides detailed description of the methods used to solve entire pipeline of the system. Chapter 4 presents the result obtained for different datasets. Chapter 5 presents the detailed discussion of the results obtained. In chapter 6 a conclusion is provided.

Chapter 2

Literature Review

2.1 Introduction

The purpose of this chapter is to provide background knowledge and review existing research related to LiDAR SLAM, bundle adjustment, and loop detection.

In mobile robotics, map is an important prerequisite that enables robots to operate and perform different tasks such as navigation, obstacle avoidance, path planning, etc [1]. SLAM is the most popular method in the literature to generate maps[2]. Among the different methods of SLAM, vision-based and LiDAR technologies are the most popular in the literature[3]. Although each SLAM method has its advantages and drawbacks, they are essential for different tasks. LiDAR technologies will likely remain essential for years to come due to their numerous advantages[4]: robust to various lighting conditions, accurate 3D measurements with higher Field of View (FoV), among others, despite their high cost and bulkiness to install in small-scale robots like drones. Solid-state LiDARs are cost-effective, lightweight, and precise, making them ideal for UAVs in mapping and complex environments[5].

Although these solid-state LiDARs have good potential, they have presented new challenges to SLAM solutions: 1) When the robot is operating in a cluttered environment, there not be strong features with geometric shapes like edges and planes. This causes the LiDAR SLAM to degenerate easily. 2) Fusing many feature points with IMU is challenging. 3) LiDAR points are sequentially sampled resulting in each point having a different timestamp which causes motion distortion that severely affects the point registration [6].

Several works exist in the literature on LiDAR SLAM[7]. Some of these works use filter-based approach [8, 9], Kalman Filter (KF) for instance, while others use optimization-based(factor graphs)[10, 11] for the state estimation. Although filter-based approaches are simple in terms of implementation and computational requirements, they accumulate errors over time causing a larger drift. Optimization-based approaches on the other hand formulate the SLAM problem as non-linear problem and optimize the whole robot trajectory and the map minimizing the drift, more specifically during loop closure. In the following sections, basic probabilistic formulations, algorithms, and concepts related to SLAM are introduced. The most related and relevant works are also reviewed.

2.2 Probabilistic notations

In order to model uncertainty in robotic systems, it is essential to define a clear probabilistic framework. The behavior of robot states, sensor measurements, and control actions is inherently stochastic due to noise and imperfections in sensing and actuation. Therefore, before presenting probabilistic formulations in later sections, we clarify the notations and concepts such as joint distribution, conditional independence and probabilities that will be used throughout. This section ensures that subsequent derivations and notations are consistent, and interpretable.

The sensor measurements, controls, and robot states are modeled as random variables. This modeling helps to correctly incorporate the uncertainties that exist due to the sensor measurements, the motion model of the robot, or the environment itself[12]. These variables can take multiple values following probabilistic laws. Probabilistic inference calculates the probability distributions of derived random variables, such as sensor data. Let $\mathbf{X}$ denote a random variable and x a possible event. The probability of $\mathbf{X}$ taking a value x is denoted as

$$p(X = x) \quad (2.1)$$

Eq (2.1) can also be written as $p(x)$ for simplicity. The sum of all possible probabilities of discrete random variable $\mathbf{X}$ sum to 1.

$$\sum_x p(x) = 1 \quad (2.2)$$

To make estimation in a continuous space, a set of continuous random variables with Probability Density Function (PDF) are used. Most commonly a normal distribution

with mean μ and variance σ^2 , shown in Eq. (2.3), is used.

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (2.3)$$

The PDF for multivariate random variable, usually expressed as vector form, is shown in Eq. (2.4) where $\mu \in \mathbb{R}^n$ is the mean and Σ is the *covariance matrix*.

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \quad (2.4)$$

To express the joint probability distribution of two random variables $\mathbf{X}$ and $\mathbf{Y}$, we use the notation $p(x, y) = p(\mathbf{X} = x \text{ and } \mathbf{Y} = y)$. If $\mathbf{X}$ and $\mathbf{Y}$ are independent, $p(x, y) = p(x) p(y)$.

The conditional probability of x given y is denoted as:

$$p(x | y) = \frac{p(x, y)}{p(y)} \quad (2.5)$$

Leveraging the conditional probability shown in Eq. (2.5), Bayes rule can be derived and put as in the following Eq. (2.6).

$$p(x | y) = \frac{p(y | x) p(x)}{p(y)} \quad (2.6)$$

It is also possible to condition the Bayes rule described in Eq. (2.6) on arbitrary other variables, z , as shown in Eq. (2.7).

$$p(x | y, z) = \frac{p(y | x, z) p(x | z)}{p(y | z)} \quad (2.7)$$

Similarly, probabilities of independent random variables x and y can be combined on other variable z as in Eq. (2.8).

$$p(x, y | z) = p(y | z) p(x | z) \quad (2.8)$$

2.3 LiDAR Odometry and LiDAR Inertial Odometry

Accurate 3D mapping with LiDAR requires efficiently registering the point clouds to a global map frame[7]. This efficiency of registration depends on accurate estimation of the LiDAR pose in continuous motion. Independent motion estimation systems such as

Global Navigation Satellite System (GNSS) or Inertial Navigation System (INS) can be used for registration, however GNSS systems do not provide continuous signal in diverse environments and INS accumulate drifts over time resulting in inconsistency in the 3D map.

LiDAR Odometry (LO) is a continuous estimation of the motion of the vehicle by analyzing successive 3D point clouds captured by LiDAR sensor mounted on it. This works by tracking the changes in the structure of the environment that is achieved aligning the current scan to the previous and computing the relative motion. LO methods are robust in environments with rich features (planes, and edges). The most widely used registration method Iterative Closest Point (ICP) is one example that can be used to estimate motion with LO.

However, LO suffers from drift and degeneration specially in challenging environments such as repetitive structures and dynamic objects. Fusing high frequency IMU data with LiDAR can rescue LO and by reducing the drift thereby improving the motion estimation. This fusion of LiDAR and IMU for motion estimation is known as LIO. Several researches [13, 14, 6] have been conducted demonstrating the effectiveness LIO for motion estimation and 3D mapping.

2.4 Lie Theory and 3D pose representation

The proper formulation of many estimation algorithms in robotics demands stability, precision, and consistency of the solutions. To achieve this, proper modeling of states and measurements, relationship between them, and uncertainties of the relation is important [15]. To this end, a smooth topological surfaces of Lie groups known as manifolds are designed. Manifolds are mathematical curved spaces that resemble euclidean space ($\mathbb{R}^n$) but they actually have more complicated global structure.

2.4.1 Lie groups

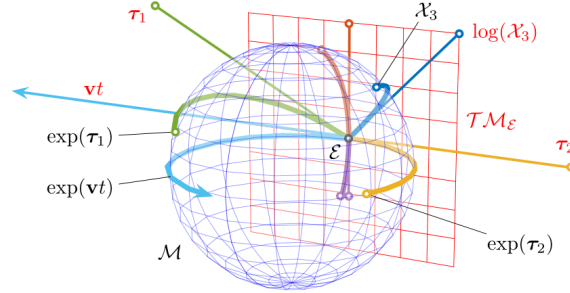


Figure 2.1: Relationship between Lie Groups and Lie algebras[15]

A Lie group is a group that is also a smooth manifold, with group operations (multiplication and inversion) that are differentiable. In robotics, Lie groups model continuous transformation groups such as rotations and rigid body motions. They enable representing and composing robot poses in a mathematically consistent way.

Orthogonal Groups and Special Orthogonal Groups

The orthogonal group $O(n)$ represents all $n \times n$ matrices that preserve length and angles. The special orthogonal group, $SO(n)$ consists of a set of all rotation matrices with determinant of $+1$. To represent 3D orientation of links, rotating sensors, to compute jacobian of motion models in robotics an $SO(3)$ is used. Eq. (2.9) shows the orthogonal groups where $GL(n)$ is a general linear group of $n \times n$ matrices with non-zero determinant.

$$\begin{aligned} O(n) &= \{ \mathbf{R} \in GL(n) : \mathbf{R}^T \mathbf{R} = \mathbf{I}_{n \times n} \} \\ SO(n) &= \{ \mathbf{R} \in O(n) : \det(\mathbf{R}) = 1 \} \end{aligned} \quad (2.9)$$

Euclidean Groups and Special Euclidean Groups

The euclidean groups, $E(n)$ and special euclidean groups, $SE(n)$, the rigid transformation(rotation, and translation) in n dimensional space, see Eq. (2.10).

$$\begin{aligned} E(n) &= \left\{ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} : R \in O(n), t \in \mathbb{R}^n \right\} \subset GL(n+1) \\ SE(n) &= \left\{ \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} : R \in SO(n), t \in \mathbb{R}^n \right\} \subset GL(n+1) \end{aligned} \quad (2.10)$$

2.4.2 Lie algebra

The Lie algebra $\mathbf{T}_{\mathbf{E}}\mathcal{M}$ (red plane) to the Lie group's manifold $\mathcal{M}$ (blue sphere) is a tangent space at the identity denoted as $\mathbf{E}$ in figure 2.1. Each element in $\mathbf{T}_{\mathbf{E}}\mathcal{M}$ has its equivalent on $\mathcal{M}$. For example, $\mathbf{v}t$ produces path $\exp(\mathbf{v}t)$, and $\log(X)$ corresponds to $\mathbf{X}$ in figure 2.1.

The group $SO(3)$ is a Lie group, while the space $\mathfrak{so}(3)$ is its corresponding Lie algebra. The latter is the tangent space at the identity of $SO(3)$. The Lie group is a complicated nonlinear object, while its Lie algebra is a vector space that is simpler to work with.

$$\begin{aligned} \exp : g &\longrightarrow G \\ \log : G &\longrightarrow g \end{aligned} \quad (2.11)$$

The operators $\boxplus$ and $\boxminus$ can express addition and subtraction for $\mathbf{X}, \mathbf{Y} \in \mathcal{M}$ and $\hat{\mathbf{u}} \in \mathbf{T}_{\mathbf{E}}\mathcal{M}$.

$$\begin{aligned} \mathbf{X} \boxplus \mathbf{u} &\doteq \text{Exp}(\mathbf{u}) \circ \mathbf{X}, \\ \mathbf{X} \boxminus \mathbf{Y} &\doteq \text{Log}(\mathbf{X} \circ \mathbf{Y}^{-1}). \end{aligned} \quad (2.12)$$

2.5 Introduction to Bundle Adjustment

Bundle Adjustment (BA) is a key optimization technique widely used in visual applications such as Structure from Motion (SfM) and Visual SLAM (V-SLAM)[16]. Its objective is to jointly refine the 3D structure of the environment and the camera poses by minimizing the reprojection error between the observed 2D feature points and the projections of the estimated 3D points onto the image plane, as illustrated in Figure 2.2. This process adjusts both the camera parameters and the 3D point positions to best fit the observations.

$$\min_{\mathbf{X}, \mathbf{P}} \sum_{i=1}^M \sum_{j=1}^N \|\mathbf{z}_{ij} - \pi(\mathbf{P}_i, \mathbf{X}_j)\|^2 \quad (2.13)$$

The mathematical formulation of the BA objective function is given Eq. 2.13 where $\mathbf{X}_j$ denotes the 3D coordinates of point j , $\mathbf{P}_i$ represents the camera pose parameters at frame i , $\mathbf{z}_{ij}$ is the observed 2D image measurement of point j in frame i , and $\pi(\cdot)$ is the projection function mapping 3D points to 2D image coordinates. One of the primary limitations of BA is its computational complexity, which is approximately $\mathcal{O}((qN + \ell M)^3)$ [16], where N is the number of 3D points, M is the number of camera poses, and q and ℓ are the number

of parameters associated with each point and pose, respectively. This high computational demand makes full BA challenging for large-scale datasets and long trajectories.

To reduce the computational cost of BA, practical systems often use windowed bundle adjustment, limiting optimization to a small set of recent frames. Alternatively, pose-only BA fixes the 3D points and optimizes only the camera poses, offering a more efficient solution while still improving accuracy..

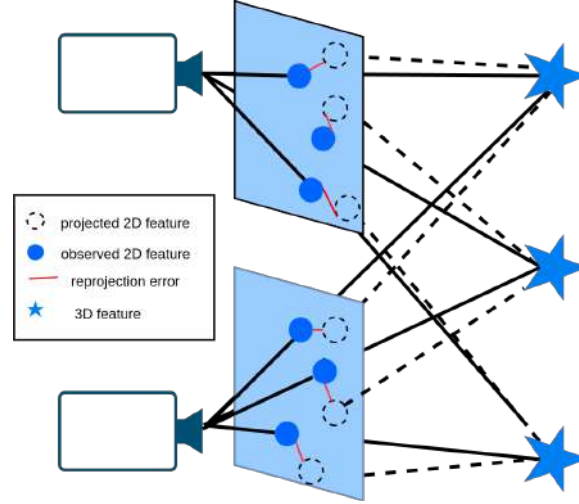


Figure 2.2: Visual BA

2.5.1 LiDAR Bundle Adjustment

In large-scale SLAM applications, such as at the city scale, map inconsistencies are common, even after PGO, since it optimizes only relative pose errors between two LiDAR frames. Additionally, loop closure errors can significantly impact the global consistency of the map [17]. To address this, LiDAR Bundle Adjustment (LBA) can be employed to improve global consistency. Furthermore, vertical misalignments, which most LIO systems are vulnerable for, can be corrected using LBA, although GPS data, if available, may also help mitigate vertical drift.

Similar to visual BA, the LBA problem can be formulated to jointly determine the LiDAR poses and the global 3D point cloud map thereby decreasing the drift caused by accumulation of scan registration errors. Even though LBA seems simpler than visual BA due to accurate depth measurements, its formulation is not trivial. Due to the sparse and non repetitive nature of LiDAR point clouds, the natural formulation used in visual BA to minimize the re-projection error does not apply to LBA [18].

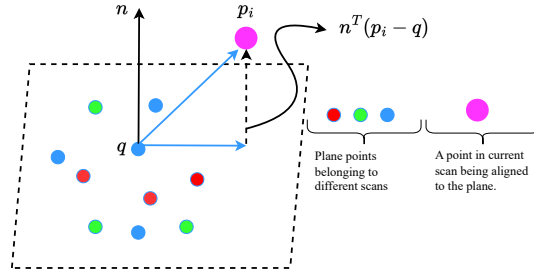


Figure 2.3: A plane in space and corresponding feature points drawn from different scans

A work by Liu et al. [19] introduced a formulation of BA on sparse LiDAR feature points with edges and planes to refine map incrementally built with LiDAR SLAM. In this formulation, unlike visual BA, the plane or edge parameters (the normal vector $\mathbf{n}$, and point in the plane $\mathbf{q}$, Eq. 2.14) are first analytically solved such that the BA problem is only related to LiDAR poses and such parameters are eliminated in each iteration of optimization [20]. This drastically reduces the cost of BA by enforcing it to only perform the so called motion-only BA without the need to optimize the 3D points.

$$\min_{\mathbf{n}, \mathbf{q}} \frac{1}{N} \sum_{i=1}^N \left(\mathbf{n}^T (\mathbf{p}_i - \mathbf{q}) \right)^2 \quad (2.14)$$

Eq. (2.15) shows the direct formulation of LBA that minimizes the sum of squared distances of each LiDAR point to the plane where $\mathbf{p}_i$ is a point observed from pose $\mathbf{T}_i$, and $\mathbf{A}$ is the covariance matrix of the observed points, λ_k is the k -th smallest eigenvalue of $\mathbf{A}$, $\mathbf{u}_k$ is the corresponding eigenvector.

$$\begin{aligned} (\mathbf{T}^*, \mathbf{n}^*, \mathbf{q}^*) &= \arg \min_{\mathbf{T}, \mathbf{n}, \mathbf{q}} \frac{1}{N} \sum_{i=1}^N \left(\mathbf{n}^T (\mathbf{p}_i - \mathbf{q}) \right)^2 \\ &= \arg \min_{\mathbf{T}} \left(\min_{\mathbf{n}, \mathbf{q}} \frac{1}{N} \sum_{i=1}^N \left(\mathbf{n}^T (\mathbf{p}_i - \mathbf{q}) \right)^2 \right) \\ &= \lambda_3(\mathbf{A}), \quad \text{if } \mathbf{n}^* = \mathbf{u}_3, \mathbf{q}^* = \bar{\mathbf{p}} \\ \text{where } \bar{\mathbf{p}} &= \frac{1}{N} \sum_{i=1}^N \mathbf{p}_i, \quad \mathbf{A} = \frac{1}{N} \sum_{i=1}^N (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^T \end{aligned} \quad (2.15)$$

The optimization on the scan poses $\mathbf{T}$ finally corresponds to minimizing the eigenvalues of matrix $\mathbf{A}$ in Eq. (2.15). The formulation defined in Eq. (2.15) requires the set of points, $\mathbf{p}_i$ acquired from set of scan poses $\mathbf{T}$, belonging to the same plane feature to be defined. Adaptive voxelization technique [19] is used to define the points that belong to the plane. If

all points in the voxel lie on the plane by examining the eigenvalue of the point covariance matrix, that voxel is kept in memory otherwise the voxel is further subdivided into eight octants until a certain minimum threshold voxel size is reached. This generates voxel map with voxels of different sizes adapted to the environment.

The voxelization process described above extends to non-planar features such as curved surface when the voxel map is performed at a finer level. The maximum depth of a tree and the minimum number of points contained in a voxel are the two conditions used to stop the recursive subdivision of a voxel.

2.6 Overview and types of SLAM

There are several types of SLAM solutions. Depending on the type of exteroceptive sensors used, LiDAR SLAM and V-SLAM are the most commonly used types. LiDAR SLAM uses laser sensors to measure distances and create precise 3D maps, making it highly effective in dark or low-feature environments. However, its high cost and power consumption make it less suitable for low-cost robotics. On the other hand, V-SLAM depends on monocular, stereo, RGB-D or event based cameras to detect visual features and track motion, providing a more affordable and lightweight solution. While V-SLAM captures detailed textures and colors, it struggles in low-light conditions and areas with few visual features. Despite these differences, both types are essential in robotics and autonomous systems. To improve accuracy, many modern applications combine both techniques using sensor fusion, taking advantage of their strengths to enhance localization and mapping performance. Based on the solution methodology used, SLAM can be further categorized into online SLAM and Full SLAM.

2.6.1 Online SLAM

Online SLAM methods estimate the current state x_t and the whole map m based on all past observations $z_{1:t}$ and controls $u_{1:t}$. These methods use KF, Extended Kalman Filter (EKF), Particle Filter (PF), and Unscented Kalman Filter (UKF)[21]. EKF-SLAM is one of the most popular online SLAM algorithms that uses Bayes filter to estimate the robot poses and landmark locations, assuming Gaussian distributions and local linearity. Bayes filter is a recursive filter for state estimation that includes two steps, prediction and update as shown in Eq (2.16), where $belief(x_t)$ represents the prior belief about the robot state

at time t before incorporating the latest observation (prediction step) and $belief(x_t)$ is a posterior belief that defines the robots refined state after incorporating latest observation (update step).

$$\begin{aligned} \bar{belief}(x_t) &= \int p(x_t | x_{t-1}, u_t) \bar{belief}(x_{t-1}) dx_{t-1}, \\ belief(x_t) &= p(z_t | x_t) \bar{belief}(x_t). \end{aligned} \quad (2.16)$$

Map generated with online SLAM is not accurate due to the linearization errors accumulated over time, local state estimation and absence of loop closure constraint to refine accumulated errors.

2.6.2 Full SLAM

FuLL SLAM aims to solve the problem of finding the most probable sequence of robot poses $x_{1:t}$ and the map m , given all sensor measurements $z_{1:t}$ and control inputs $u_{1:t}$ [12]. It is mostly used to build consistent map using point clouds and solve SfM problems, specially to handle loop closure events.

Bayesian network

One approach to solving the Full SLAM problem is using Bayesian network where the problem is formulated as joint distribution of all the variables ($x_{1:t}$, $z_{1:t}$, $u_{1:t}$, and m). It is a probabilistic directed graph model for solving full SLAM where the entire problem is formulated as joint distribution of the robot poses ($x_{1:M}$), the measurements ($z_{1:K}$), controls ($u_{1:M}$), and landmarks ($l_{1:N}$). The joint probability distribution of the whole SLAM is defined as in Eq. (2.17) where the variables X , L , U , and Z are all robot poses, all landmarks, all control inputs, and all measurements respectively Eq.(2.18). Figure 2.4 shows a typical Bayes network where every node encodes a conditional probability on the variable and its parent nodes.

$$P(X, L, U, Z) \propto P(\mathbf{x}_0) \prod_{i=1}^M P(\mathbf{x}_i | \mathbf{x}_{i-1}, \mathbf{u}_i) \prod_{k=1}^K P(\mathbf{z}_k | \mathbf{x}_{i_k}, \mathbf{l}_{j_k}) \quad (2.17)$$

$$\begin{aligned} X &= \{\mathbf{x}_i\}, & i &\in 0 \dots M \\ L &= \{\mathbf{l}_j\}, & j &\in 1 \dots N \\ U &= \{\mathbf{u}_i\}, & i &\in 1 \dots M \\ Z &= \{\mathbf{z}_k\}, & k &\in 1 \dots K \end{aligned} \quad (2.18)$$

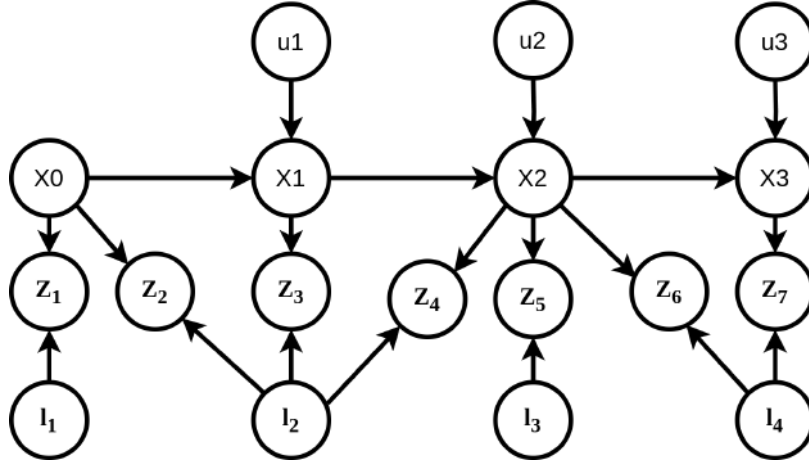


Figure 2.4: Probabilistic Bayesian net SLAM model

Factor Graph

Bayesian network uses only arrows to define dependency between variables which does not encode much of information. Effortlessly, Bayesian network can be converted to factor graphs by conditioning on sensor data [22]. Figure 2.5 shows the factor graph obtained by converting the Bayesian network shown in figure 2.4. Each node in the Bayesian network is split in both variable node and factor node in the graph. The factor is connected to a variable node and parent variable nodes in the graph.

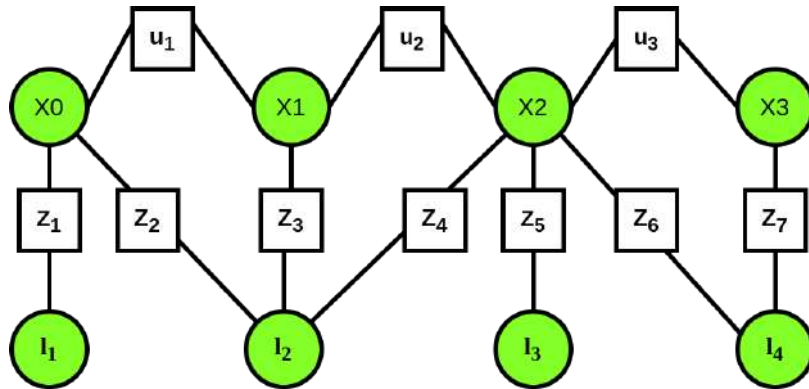


Figure 2.5: Full SLAM model with factor graph

Factor Graphs represent the probability distribution as a bipartite graph with variable nodes (green circles in Fig. 2.5) and factor nodes (white squares in Fig. 2.5) where edges can only connect variable nodes via factor nodes. These factor nodes encode information that enters into the system and the graph captures the propagation of these information to the hidden states that we intend to estimate. We denote variable nodes as $\mathbf{x}$ and factor

nodes as $\emptyset$. The factors define the probabilistic constraint motion model (ϕ_i) and the likelihoods measurements (ϕ_k). The joint probability distribution of the problem is then computed as the product of all factors in the graph Eq. (2.19). This graph structure allows efficient optimization via methods such as Gaussian Belief Propagation or Nonlinear Least Squares Optimization. Factor graph is the core of the prevalent SLAM technique, PGO, that models each factors as Gaussian distributed pose constraints on the SE3 manifold. It is in practice difficult to accurately estimate the covariance matrix of this Gaussian Distributed constraints. It is commonly estimated using the Hessian matrix of ICP alignment [23], and Monte-Carlo-based estimation[24], however the former is optimistic and the later needs to perform scan matching multiple times. Most of the LiDAR SLAM frameworks [20, 14, 25, 26], use only constant covariance matrix and weighting scheme.

$$p(\mathbf{X} | \mathbf{Z}) \propto \prod_{k=1}^K \phi_k \quad \text{where } \phi_k \geq 0 \quad \forall K \quad (2.19)$$

Solving the SLAM problem with factor graph is exactly determining the values of the unknown variable nodes($\mathbf{X}$) that maximize the agreement with the uncertain measurements($\mathbf{Z}$). This is known as maximum a posteriori estimate (MAP). Solving for the states (variable nodes) that maximize the posterior $p(X | Z)$ Eq. (2.20) can be performed by exploiting non-linear optimization methods.

$$X^{MAP} = \arg \max_X p(X|Z) = \arg \max_X \frac{p(Z|X)p(X)}{p(Z)} \quad (2.20)$$

2.7 Related works

Ji Zhang et al. [27] proposed a real-time method for LiDAR odometry and mapping (LOAM). They achieved low-drift results without the need to use IMU and high-accuracy range measurements. In their method, the costly SLAM problem is divided into to algorithms where one algorithm computes the odometry at a higher frequency and the other algorithm runs at a much lower frequency to register point cloud. While it achieves impressive accuracy and real-time performance, it lacks inertial fusion and loop closures are not considered.

A method by Xu et al. [6] presented a novel approach to LIO called *Fast, Robust LiDAR Inertial Odometry package by tightly-coupled Iterated Kalman Filter* (FAST-LIO)

by proposing a new formula for computing the Kalman gain, which demonstrates equivalence to the conventional formula but reduces computation complexity based on state dimensions rather than measurement dimensions. An Iterated Extended Kalman Filter (IEKF) is used to minimize the point-to-plane distance during the registration of scan to map. The package operates effectively on a small-scale quadrotor with fast motion resulting in real-time performance and stable odometry output and mapping at a frequency of 50HZ. Experiments conducted in varied environments validate the system's resilience against fast motion and vibration noise. Despite its resilience to fast motion and cluttered environments, FAST-LIO does not include loop closure or global pose graph optimization causing accumulation of drift over long trajectories. Hence, in our research, we intended to include loop closure constraint on top of FAST-LIO to correct accumulated drifts over time. To this end, we conducted a review of available loop detection methods and present the relevant and robust method.

Gupta et al. [28] proposed a robust loop closure detection pipeline for LiDAR-based SLAM that effectively handles variations in LiDAR sensors, including different scanning patterns, fields of view, and resolutions. Their approach generates local maps from consecutive LiDAR scans and employs a ground alignment module to ensure consistency across diverse motion profiles. A key feature of their method is the use of density-preserving Bird's Eye View (BEV) projections, from which ORB feature descriptors are extracted for efficient place recognition. These descriptors are stored in a binary search tree, enabling fast retrieval while mitigating perceptual aliasing through self-similarity pruning.

Kim et al. [29] presented Scan Context, a novel spatial descriptor designed for outdoor place recognition using 3D LiDAR scans. Unlike traditional histogram-based methods [30, 31, 32], Scan Context encodes the entire point cloud into a matrix format, preserving the absolute geometrical structure of the environment. This approach utilizes a cosine distance measure for similarity scoring and implements a two-phase search algorithm to enhance loop detection efficiency, particularly in urban settings. Experimental results demonstrate that Scan Context significantly outperforms existing global descriptors in various datasets, providing robust performance against noise and viewpoint changes.

A work by Xiyuan Liu et al. [20] presented a novel approach for large-scale LiDAR mapping by introducing a HBA framework. The method aims to improve the consistency

and accuracy of LiDAR-based maps by formulating a global optimization problem that jointly refines multiple scans. Unlike conventional SLAM techniques that rely on pairwise constraints, HBA leverages a hierarchical structure to balance local precision and global consistency, mitigating drift accumulation over extended trajectories. The authors demonstrate the effectiveness of their approach through extensive experiments on real-world datasets, showcasing its advantages in handling large-scale environments with minimal loop closures. Their results indicate that HBA outperforms existing methods in terms of mapping accuracy and robustness, particularly in scenarios with sparse revisits. A key contribution of this work is the introduction of an efficient optimization strategy that scales well with the number of LiDAR scans. By organizing the LiDAR data into hierarchical layers, the proposed framework enables efficient refinement without requiring excessive computational resources. The authors also incorporate a robust outlier rejection mechanism to enhance data association, ensuring reliable pose estimation in challenging conditions. Furthermore, the method integrates seamlessly with existing LiDAR-based odometry pipelines, making it adaptable to various robotic platforms. However, HBA map consistency degrades if there are more revisits and loop closures especially if the divergence in the map due to drift is larger than the maximum voxel size, BA might have slow convergence rate.

Wei Xu et al. [13] significantly improved upon the original work [6] by introducing two key advancements: direct point registration and the incremental k-D tree (*ikd-Tree*). Unlike [6], which relies on hand-crafted feature extraction, the new approach registers raw LiDAR points directly to the map, increasing accuracy and adaptability to different LiDAR sensors. The newly introduced *ikdTree* enables efficient incremental updates, dynamic re-balancing, and on-tree downsampling, reducing computational load while maintaining real-time performance. These improvements allow a new approach to achieve higher accuracy at a lower computational cost, making it more robust for diverse environments, including those with small FoV LiDARs and aggressive motion. Compared to their original work, the overall processing time for a single scan was improved by approximately a factor of ten.

Chapter 3

Methodology

3.1 System Overview

The proposed system integrates FAST-LIO2 [13] for odometry, Scan Context for loop closure detection, PGO using Georgia Tech Smoothing and Mapping (GTSAM [33]) for correcting drifts in the odometry, and finally HBA for global refinement of the map. The diagram shown in Figure 4.1 illustrates the integrated components used in system pipeline where the LIO and PGO can run in real-time but due to the computational and memory cost, HBA is run offline.

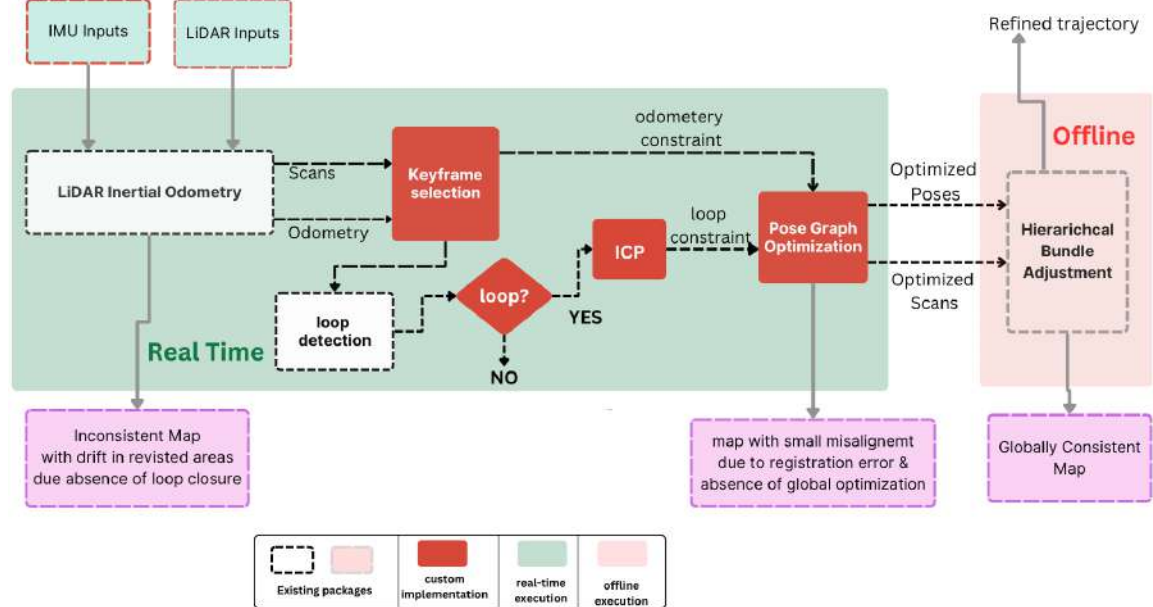


Figure 3.1: Block diagram of the proposed system pipeline.

Algorithm 1 LiDAR-Inertial Odometry with Global Optimization

```

1: Input:  $\mathcal{I}$  (IMU data),  $\mathcal{L}$  (LiDAR scans)
2: Output:  $\mathcal{M}^*$  (optimized map),  $\mathcal{T}^*$  (refined trajectory)
3: Initialize:  $\mathcal{K}$ (Keyframe set)  $\leftarrow \emptyset$ ,  $\mathcal{G}$ (Bipartite graph)  $\leftarrow \emptyset$ 
4: while new  $(\mathcal{I}_t, \mathcal{L}_t)$  available do
5:    $\hat{\mathbf{T}}_t \leftarrow \text{LIO}(\mathcal{I}_t, \mathcal{L}_t)$ 
6:   if ISKEYFRAME( $\hat{\mathbf{T}}_t$ ) then
7:      $\mathcal{K} \leftarrow \mathcal{K} \cup \{k_t\}$ 
8:      $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{C}_{\text{odom}}(k_t)\}$ 
9:     if DETECTLOOP( $k_t, \mathcal{K}$ ) then
10:       $\mathcal{G} \leftarrow \mathcal{G} \cup \{\mathcal{C}_{\text{loop}}(k_t, k_l)\}$ 
11:       $\{\mathbf{T}_i^*\} \leftarrow \text{OPTIMIZEPOSEGRAPH}(\mathcal{G})$ 
12:    end if
13:  end if
14: end while
15:  $\{\mathbf{T}_i^{**}, \mathcal{M}^*\} \leftarrow \text{HBA}(\{\mathbf{T}_i^*\}, \mathcal{K})$ 
16: return  $\mathcal{M}^*$ ,  $\mathcal{T}^* = \{\mathbf{T}_i^{**}\}$ 

```

FAST-LIO2 outputs odometry and updated LiDAR scan frames at a frequency of 10-100Hz. The scan frames are updated according to the state estimation module that includes prediction and update steps. Aggregating these scans using the odometry poses can generate accurate map for short trajectories because LIO performs well for sequential registration of scans. However after long trajectory and revisiting the same locations, the map diverges due to accumulated drifts in the odometry.

In the subsequent sections, detailed description of each of the components used in the system will be presented.

3.2 Description of FAST-LIO2

FAST-LIO2 is a robust and computationally efficient LiDAR-inertial odometry system that tightly fuses LiDAR feature points with IMU data using an IEKF. This enables reliable navigation even in challenging scenarios involving fast motion, sensor noise, or cluttered environments where traditional methods may fail due to degeneracy. To output odometry at higher frequency, a state estimation module fuses both IMU and LiDAR inputs with an IEKF. This involves prediction of the state based on IMU input only also called forward propagation, then iteratively update it with LiDAR inputs until the residual drops below a certain threshold. The state vector $\mathbf{x}$ is a 24-dimensional vector

containing the state variables shown in Eq. (3.1).

$$\begin{aligned}
 \mathbf{x} &\triangleq ({}^G\mathbf{R}_I^T, {}^G\mathbf{P}_I^T, {}^G\mathbf{V}_I^T, \mathbf{b}_\omega^T, \mathbf{b}_a^T, {}^G\mathbf{g}^T, {}^I\mathbf{R}_L^T, {}^I\mathbf{P}_L^T)^T \in \mathcal{M} \\
 \mathcal{M} &\triangleq SO(3) \times \mathbb{R}^{15} \times SO(3) \times \mathbb{R}^3, \quad \dim(\mathcal{M}) = 24 \\
 \mathbf{u} &\triangleq [\boldsymbol{\omega}_m^T \ \mathbf{a}_m^T]^T, \quad \mathbf{w} \triangleq [\mathbf{n}_\omega^T \ \mathbf{n}_a^T \ \mathbf{n}_{b\omega}^T \ \mathbf{n}_{ba}^T]^T
 \end{aligned} \tag{3.1}$$

${}^G\mathbf{R}_I^T$, ${}^G\mathbf{P}_I^T$, and ${}^G\mathbf{V}_I^T$ denote orientation, position, the linear velocity of IMU in the global frame respectively. The $\mathbf{b}_\omega^T$ is the bias in angular velocity, $\mathbf{b}_a^T$ is bias in acceleration of IMU, and ${}^G\mathbf{g}^T$ is gravity. ${}^I\mathbf{R}_L^T$, and ${}^I\mathbf{P}_L^T$ denote the extrinsic calibration between IMU and LiDAR sensors. The vector $\mathbf{u}$ represents IMU inputs, and $\mathbf{w}$ denotes the noise vector.

3.2.1 Prediction Step

The continuous model is discretized at the IMU sampling period Δt and the state is estimated using the forward propagation as shown in Eq. (3.2) where the function $\mathbf{f}$ is the state transition model as described in [13]. In the forward propagation, prediction step runs at frequency of $\frac{1}{\Delta t}$ until a (next) LiDAR scan comes in. The state and covariance matrix are propagated with the process noise $\mathbf{w}$ set to zero as in Eq. (3.3) where Q_i is the noise covariance, $\mathbf{F}_{\bar{\mathbf{x}}_i}$ and $\mathbf{F}_{\mathbf{w}_i}$ are jacobians of $\mathbf{f}$ with respect to state and noise respectively.

$$x_{i+1} = x_i \boxplus (\Delta t f(x_i, u_i, w_i)) \tag{3.2}$$

$$\begin{aligned}
 \hat{\mathbf{x}}_{i+1} &= \hat{\mathbf{x}}_i \boxplus (\Delta t \mathbf{f}(\hat{\mathbf{x}}_i, \mathbf{u}_i, \mathbf{0})); \quad \hat{\mathbf{x}}_0 = \bar{\mathbf{x}}_{k-1}, \\
 \hat{\mathbf{P}}_{i+1} &= \mathbf{F}_{\bar{\mathbf{x}}_i} \hat{\mathbf{P}}_i \mathbf{F}_{\bar{\mathbf{x}}_i}^T + \mathbf{F}_{\mathbf{w}_i} \mathbf{Q}_i \mathbf{F}_{\mathbf{w}_i}^T; \quad \hat{\mathbf{P}}_0 = \bar{\mathbf{P}}_{k-1},
 \end{aligned} \tag{3.3}$$

3.2.2 Update Step

The update step corrects the predicted state by minimizing point-to-plane residuals between LiDAR points and the map. The residual z_i^k of each LiDAR point ${}^L\mathbf{p}_i$ is computed as its distance to a nearest local plane in the map. This residual Eq. (3.4) quantifies the alignment error at the k th iteration.

Since the observation contains large number points ($\approx 100\text{K}$ points), inverting a matrix in the dimension of observation is costly. Hence a new formulation Eq. (3.5) of the kalman gain $\mathbf{K}$ that needs to invert a matrix in the dimension of state is derived. Using the

residual $\mathbf{z}_k^\kappa$ and the $\mathbf{K}$, the predicted state is updated as shown in Eq. (3.6).

$$\mathbf{z}_j^\kappa = \mathbf{u}_j^T \left({}^G\hat{\mathbf{T}}_{I_k}^\kappa {}^I\hat{\mathbf{T}}_{L_k}^\kappa {}^L\mathbf{p}_j - {}^G\mathbf{q}_j \right), \quad (3.4)$$

$$\mathbf{K} = \left(\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} + \mathbf{P}^{-1} \right)^{-1} \mathbf{H}^T \mathbf{R}^{-1}, \quad (3.5)$$

$$\hat{\mathbf{x}}_k^{\kappa+1} = \hat{\mathbf{x}}_k^\kappa \boxplus \left(-\mathbf{K} \mathbf{z}_k^\kappa - (\mathbf{I} - \mathbf{K} \mathbf{H})(\mathbf{J}^\kappa)^{-1}(\hat{\mathbf{x}}_k^\kappa \boxminus \hat{\mathbf{x}}_k) \right). \quad (3.6)$$

Sequential state estimation with LIO suffers from drift over long trajectories due to inherent noise in the robot motion, sensor measurements, data association problems, or dynamics in the environment [28]. Due to such a drift, when the robot revisits the same place, its belief may indicate a different location than it already knows. To overcome this problem, we proposed to integrate PGO with LIO so that such accumulated errors are corrected. PGO adds loop closure constraints to the robot poses when the same locations are revisited. To add loop closure constraints, a methodology to efficiently detect loop closure is required. In the next section, we discussed a robust loop closure detection methods and elaborated on one method we used in our research.

3.3 Loop Closure Detection

To maintain accurate pose estimate by mitigating global drift, loop closure via recognizing revisited places plays a vital role in SLAM. Therefore, a robust and efficient methodology for detecting loop closures is required to enforce a strong constraint in the graph. This enables us to have not only a better estimate of the current state and better model of the environment but also a better estimate of where the robot was in the past, since all the past poses will be updated as well.

Even though having many loop closure constraints in the graph improves our estimate, special attention should be paid not include false positive constraints by incorrect association or perceptual aliasing in the graph because they will introduce a large error into the our estimation and it is difficult to remove them once they are added. It is better to miss a true loop closures than it is to add a false one. It is important to emphasize that loop closure relies on absolute measurements of the environment, typically obtained from sensors such as LiDAR, cameras, or other environmental sensing devices. Relative sensors, including the IMU or wheel encoders, are insufficient for performing loop closure. Therefore, using these relative sensors for detecting loops should be avoided.

3.3.1 Scan Context

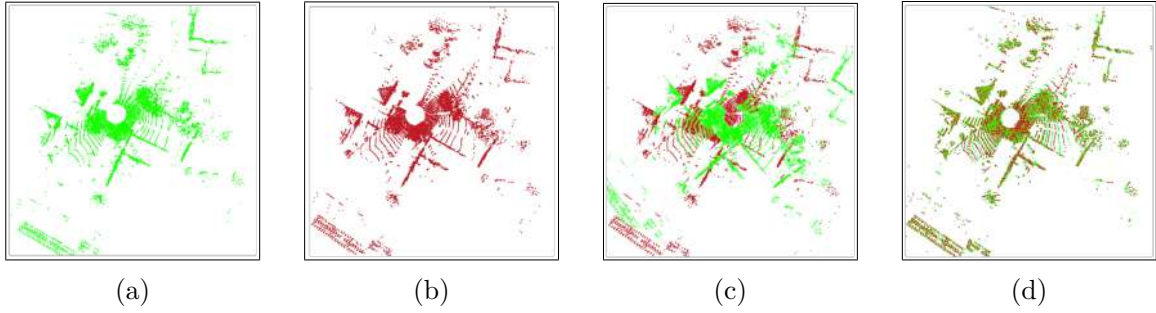


Figure 3.2: Loop detected with Scan Context. a) current scan(source), b) target scan in the map, c) before alignment d) after alignment with ICP

As described in 2.7, Scan Context is a novel spatial descriptor and matching algorithm designed for outdoor place recognition using a single 3D scan. It encodes the entire point cloud from a 3D scan into a 2D matrix representation of size $N_r \times N_s$ (figure 3.3) where N_r denotes the number of rings and N_s is the number of sectors. These two variables parameters that can be tuned based on the environment. The radial resolution of the rings for LiDAR with maximum range set to L_{max} is $\frac{L_{max}}{N_r}$ and the angle of one sector is $\frac{2\pi}{N_s}$.

A bin is represented as a set of points that belong to i th ring and j th sector of the 2D partition where $i \in [1, N_r]$ and $j \in [1, N_s]$. For each bin, the maximum height (z value) of points falling into the bin is stored. This way, an intensity image also known as *Scan Context* of the point cloud in a single scan is generated as in Eq. (3.8) where ϕ is a function that assigns the maximum height of the points in the given bin represented by i and j .

Detecting loop closure requires measuring the similarity between scan contexts I_q and I_c of the query and candidate scan point clouds. This similarity score is the sum of the cosine distance between columns of same index in the scan contexts, c_j^q and c_j^c , divided by N_s for normalization Eq.(3.7). In our implementation, we used values between **0.1** to **0.13** for $d(I^q, I^c)$ to make very strict loop closure detection to avoid false positive loops and outliers. Using a smaller value for this parameter may lead to missing important loop closures. Conversely, using a larger value can introduce false loops, which increases map divergence and the size of the graph, ultimately degrading performance. Therefore, careful tuning is essential to a balance between avoiding missed loops and preventing the inclusion of false ones. For more flexible and less strict values of the similarity score, we

propose a further fitness score check using ICP or similar methods to enforce robustness.

$$d(I^q, I^c) = \frac{1}{N_s} \sum_{j=1}^{N_s} \left(1 - \frac{c_j^q \cdot c_j^c}{\|c_j^q\| \|c_j^c\|} \right) \quad (3.7)$$

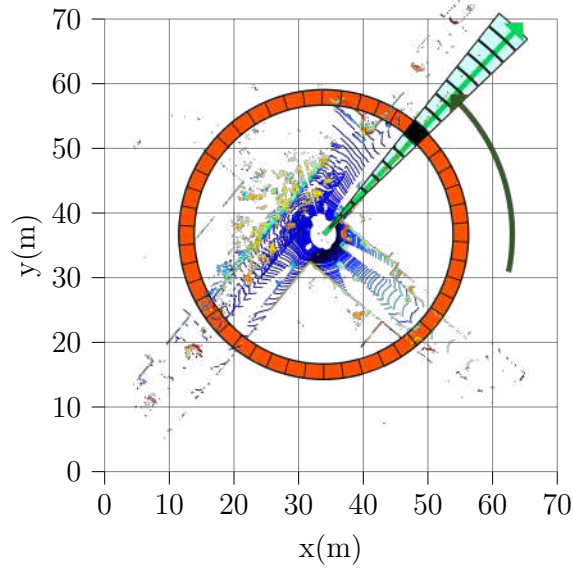


Figure 3.3: A 2D bin encoding of the 3D point cloud in a single scan[29]

The column-wise similarity scoring of scan contexts is effective for dynamic objects, however columns in the scan context for the same place can be shifted due to a change in the view point of the LiDAR sensor. To alleviate this problem, distance between query scan context and all possible column-shifted scan contexts from the original candidate scan context is computed and the shift number n that gives the minimum distance is used as initial guess for alignment Eq.(3.9).

$$I = (a_{ij}) \in \mathbb{R}^{N_r \times N_s}, \quad a_{ij} = \phi(\mathcal{P}_{ij}) \quad (3.8)$$

$$\begin{aligned} D(I^q, I^c) &= \min_{n \in [N_s]} d(I^q, I_n^c), \\ n^* &= \arg \min_{n \in [N_s]} d(I^q, I_n^c) \end{aligned} \quad (3.9)$$

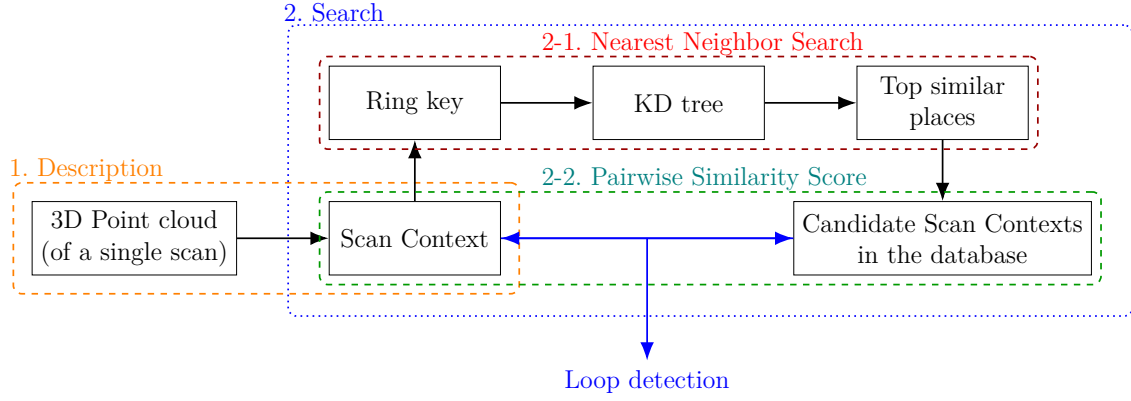


Figure 3.4: A point cloud from a single 3D scan is encoded into a Scan Context. An N_r dimensional ring key is then extracted and used to build a KD-tree for retrieving nearest candidates. These candidates are compared with the query, and the closest one that meets the acceptance threshold is selected as a loop closure[29].

Algorithm 2 Loop Detection with Scan Context

```

1: Input: Current descriptor  $I_q$ , current key  $k_q$ 
2: Output: Loop ID  $n^*$  and yaw difference  $\Delta\theta$ 
3: if number of stored descriptors  $< N_{\text{exclude}} + 1$  then
4:   return  $(-1, 0.0)$  [avoid loop with recent scans]
5: end if
6: if tree update required (every  $T$  iterations to save computation) then
7:   Exclude recent  $N_{\text{exclude}}$  keys from search set
8:   Build KD-tree on historical keys
9: end if
10: Use KD-tree to retrieve top- $K$  candidates  $\{k_n\}$  closest to  $k_q$ 
11: for each candidate  $n \in \{k_n\}$  do
12:   Compute pairwise scan context distance  $d(I_q, I_n)$  with optimal  $\Delta\theta$ 
13: end for
14: Select best match using 3.9
15: if  $D^* < \tau_{\text{SC}}$  then
16:   return  $(n^*, \Delta\theta)$ 
17: else
18:   return  $(-1, 0.0)$ 
19: end if
    
```

The number of top candidate scan contexts K in algorithm 2 is determined by the user and the candidate with the minimum distance to the query descriptor satisfying the acceptable threshold, τ_{SC} , is chosen as revisited place. Figure 3.2 shows the query(green) and candidate(red) scan satisfying the acceptable similarity threshold aligned with the initial guess $\Delta\theta$, and ICP.

3.4 ICP for computing loop closure constraint

ICP [34] is a widely used algorithm for rigid registration of 3D point clouds. It operates by iteratively refining the alignment between a source and a target point cloud by minimizing the sum of squared distances between corresponding points. In each iteration, it first identifies the closest points between the two clouds, estimates the best-fit rigid transformation (rotation and translation), applies this transformation to the source, and repeats the process until convergence. The optimal rotation $\mathbf{R}$ is computed using Singular Value Decomposition (SVD) [35] of the cross covariance matrix of the $\mathbf{H}$ shown in Eq. (3.11) of the corresponding points where x_0 and y_0 are the weighted mean of point clouds X , and Y respectively. The basic alignment problem of point sets X and Y is shown in Eq. (3.10) where x_i and y_j are corresponding points that belong to 3D point clouds X and Y respectively.

$$(\mathbf{R}^*, \mathbf{t}^*) = \arg \min_{\mathbf{R}, \mathbf{t}} \sum_{(i,j) \in \mathcal{C}} \|\mathbf{y}_i - \mathbf{R}\mathbf{x}_j - \mathbf{t}\|^2 \quad (3.10)$$

$$\mathbf{H} = \sum (y_i - y_0)(x_j - x_0)^\top, \quad \mathbf{UDV}^\top = \text{SVD}(\mathbf{H}), \quad \mathbf{R}^* = \mathbf{UV}^\top, \quad \mathbf{t}^* = \mathbf{y}_0 - \mathbf{R}\mathbf{x}_0 \quad (3.11)$$

Table 3.1: Parameters for ICP (Point-to-Point)

Symbol	Description	Value
$d_{\max}$	150 m	Maximum correspondence distance
N_{iter}	50	Maximum number of iterations
ϵ_{trans}	1×10^{-6}	Minimum transformation epsilon (convergence)
ϵ_{fit}	1×10^{-6}	Euclidean fitness epsilon (convergence)
N_{RANSAC}	0	RANSAC iterations for outlier rejection
τ_{RANSAC}	0.05 m	RANSAC outlier rejection threshold

Point Cloud Library (PCL), among others, provides implementation for several variants of ICP algorithm. We used the point-to-point implementation of ICP from the PCL library. After a pair of scans corresponding to the detected loop are found, we run the PCL’s ICP with the parameters shown in table 3.1. It is important to tune these parameters properly for optimal alignment of the points clouds in the detected scans that correspond to loops. However, in our method, since the loop closure detection module robustly detects loops with $\Delta\theta$ as initial guess, these parameters do not require excessive tuning. For short-range loop closure and indoor environments, tuning these parameters is required accordingly. The transformation ($\mathbf{R}^*$ and $\mathbf{t}^*$) that aligns the two scans corresponding to the loop is added as *loop constraint* to the factor graph.

3.5 Optimization with Pose Graph

By leveraging odometry constraints derived from LIO and loop closure constraints obtained through ICP alignment, we construct a pose graph that is subsequently optimized (see algorithm 1) to minimize a global error function shown in Eq. (3.13). In this framework, odometry constraints encode the relative motion estimates (transformation) between temporally adjacent poses, while loop closure constraints introduce corrections by identifying correspondences between non-consecutive poses, thereby mitigating accumulated drift and enhancing global consistency. Minimization of the global error function constructs configuration of the robot poses that best satisfies the constraints [36].

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j). \quad (3.12)$$

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \underbrace{\mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}}_{\mathbf{F}_{ij}} \quad (3.13)$$

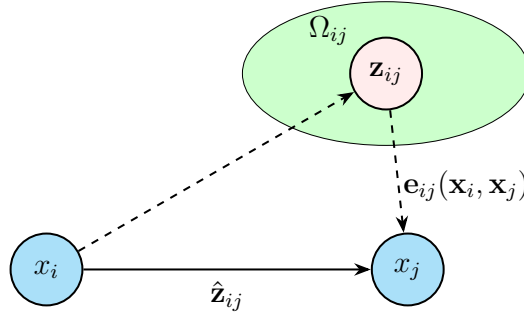


Figure 3.5: Error between actual measurement $\mathbf{z}_{ij}$ and expected measurement $\hat{\mathbf{z}}_{ij}$

For two nodes i and j in the pose graph, the function that computes the difference between the actual measurement $\mathbf{z}_{ij}$ and expected measurement $\hat{\mathbf{z}}_{ij}$ is denoted as in Eq. (3.12) [36]. For a set $\mathcal{C}$ of pairs of indices which have a constraint (observation) $\mathbf{z}_{ij}$ with information matrix $\boldsymbol{\Omega}_{ij}$, maximizing the PDF Eq. (2.20) corresponds to finding the configuration of nodes $\mathbf{x}^*$ that minimizes the negative log likelihood $\mathbf{F}(\mathbf{x})$ of all observations as in Eq. (3.14). However, in our method, we used Incremental Smoothing and Mapping (ISAM2[37]) due to its robustness, speed and suitability for real-time SLAM.

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathbf{F}(\mathbf{x}). \quad (3.14)$$

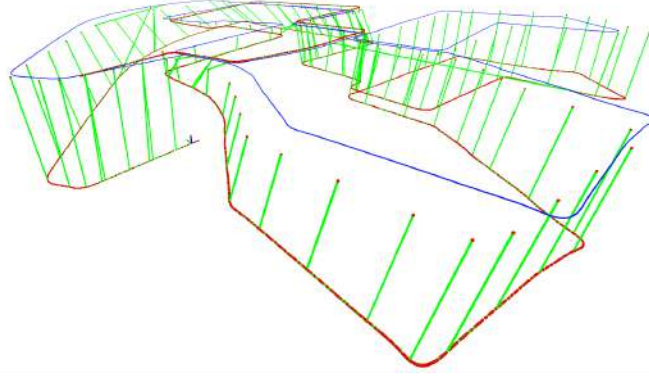


Figure 3.6: Pose graph for KITTI00 sequence before optimization, nodes are shown in red dots, and factors are shown in green lines. The blue trajectory shows the path after the pose graph is optimized.

The solution to Eq. (3.14) can be found utilizing popular optimization methods such as Gauss-Newton [38] or the Levenberg-Marquardt (LM) [39] algorithms.

3.6 Global refinement using Bundle Adjustment

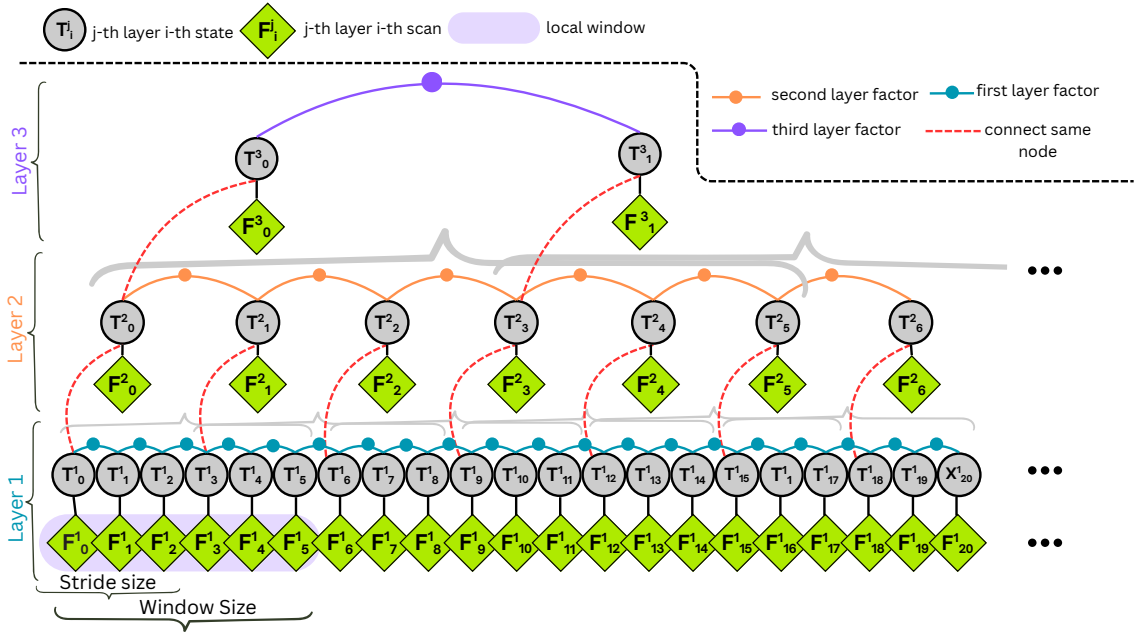


Figure 3.7: Pyramid structure of the HBA with layer number $l=3$, stride size $s=3$ and window size $w=6$ [20].

Minimization of the global error function Eq.(3.13) generates best estimate of the robot poses that best fit the measurements under Gaussian assumption of the relative pose estimation. This minimization corrects the drift in the LiDAR odometry and inconsistency

in the map. However, due to loop closure inaccuracy and optimization of only relative pose, there would still be small misalignments in the map after PGO. As discussed in section 2.5.1, LBA works to minimize the overall point-to-plane distance to optimize the mapping consistency. However this optimization process requires $\mathbf{O}(N^3)$ time to solve where N is the number of LiDAR poses involved in the computation. To address this issue, Xiyuan Liu et al. [30] proposed HBA method that globally optimizes the map consistency maintaining the efficiency of time consumption.

The architecture of HBA, as shown in figure 3.7, constructs pyramid structure of the frame poses. The refinement process involves two key steps. A bottom-up approach that performs BA within local window hierarchically from bottom layer(local BA) to the top layer(global BA). Since the number frame poses in each local window is small, this approach of employing local BA improves the time complexity by leveraging parallel computation for each local BA. However, this bottom-up process does not take into account the features that are common across different local windows and this causes inaccuracy. To mitigate this issue, a top-down process that constructs PGO from top to bottom layers and distributes the errors is used.

3.6.1 Bottom-up Process

For j -th LiDAR frame $\mathbb{F}_j^i$ in the i -th layer and its pose $\mathbf{T}_j^i = (\mathbf{R}_j^i, \mathbf{t}_j^i)$, with $\mathbf{R}_j^i \in SO(3)$ and $\mathbf{t}_j^i \in \mathbb{R}^3$, its points are represented in the local LiDAR frame and $\mathbf{T}_j^i$ is in global frame. A local BA is performed on the frames in each local window of size w and stride s in the bottom-up process given that the initial pose trajectory is provided. This local BA optimizes the relative poses between the first frame and the other frames within a local window. After optimization, a new key frame for the next layer Eq. (3.15) is created for each window by aggregating all the frames inside it. The pose of this key frame, Eq. (3.17), is computed by sequentially multiplying the optimized poses of the individual LiDAR frames in the window. This process is repeated from the bottom layer to top layer of the pyramid structure. This local BA forms relative constraints(factors) between the frames within the window. For keyframes that are appear in different windows, multiple relative constraints from different windows will constructed.

$$\mathbf{F}_j^{i+1} \triangleq \bigcup_{k=0}^{w-1} \left(\mathbf{T}_{s,j,s,j+k}^{i*} \cdot \mathbf{F}_{s,j+k}^i \right) \quad (3.15)$$

The total time consumption of HBA depends on the number of layers l , total number of LiDAR frames N , window size w , stride number s , and the number of threads n used for parallel processing. The optimal number of layers l^* of the pyramid is obtained as in Eq.(3.16).

$$l^* = \left\lfloor \frac{1}{2} \log_s \left(\frac{3N^2(s^3 - s)n}{w^3} \right) \right\rfloor \quad (3.16)$$

$$\mathbf{T}_j^{i+1} = \mathbf{T}_{s,j}^{i*} = \prod_{k=1}^j \mathbf{T}_{s,k-s,s,k}^{i*} \quad (3.17)$$

3.6.2 Top-Down Process

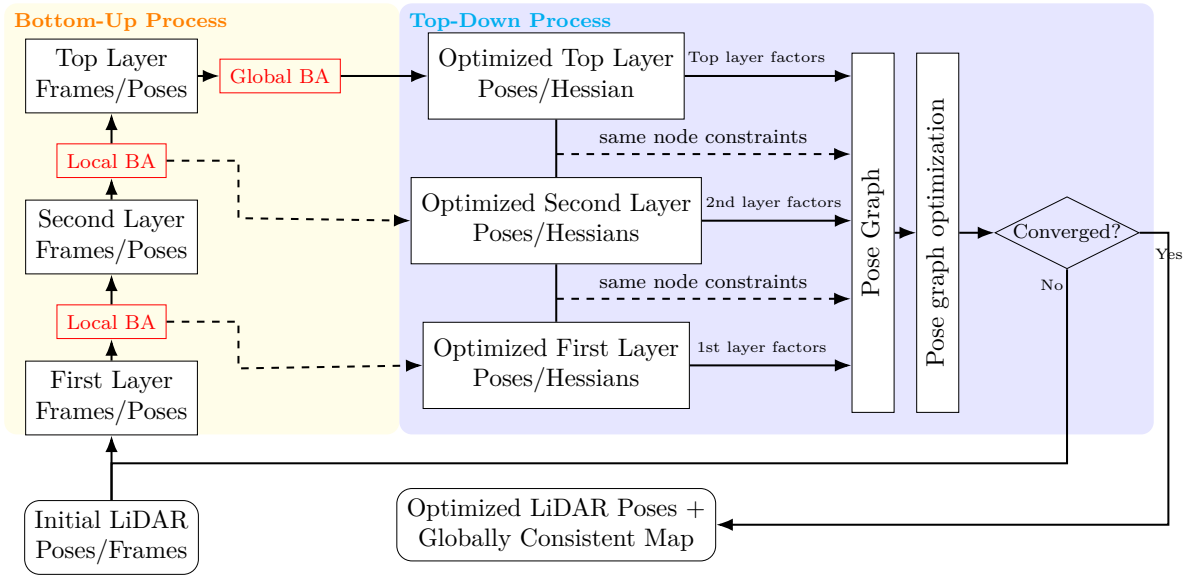


Figure 3.8: HBA optimization: Bottom-Up and Top-Down Process[20]

In the top-down process, see figure 3.8, pose graph is constructed using the frame poses as variables and relative pose between adjacent frames as factors in each layer of the pyramid. Finally PGO is used to solve the pose graph by LM using GTSAM.

Chapter 4

Experiments and Results

This chapter presents the evaluation of the proposed mapping system through a series of experiments conducted on both custom and public datasets. The custom dataset includes trajectories of varying lengths (long and short) captured in diverse settings, containing both indoor and outdoor environments. The aim is to assess the system's performance in terms of trajectory accuracy, loop closure effectiveness, map consistency, and computational efficiency.

4.1 Experimental Setup

4.1.1 Hardware and Platform

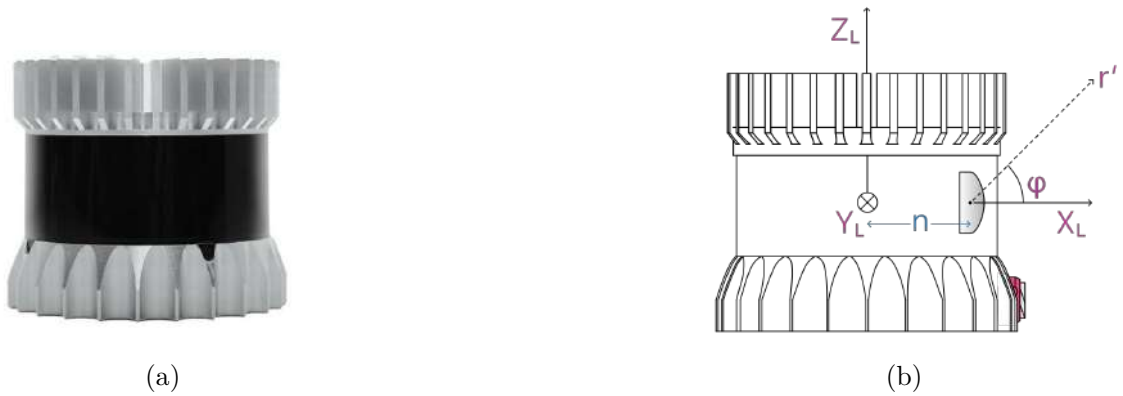


Figure 4.1: a) Ouster OS1-128 LiDAR with built-in IMU. b) Sensor coordinate frame [40]

The custom datasets were collected using the AgileX Hunter 2.0 mobile robot equipped with an Ouster OS1-128 LiDAR. The sensor provides 128 scan lines, a 360° horizontal field of view, and operates at 10 Hz. It also includes a built-in IMU, which was used

for LIO odometry estimation. The extrinsic calibration between the ouster LiDAR and the IMU can be computed online with FAST-LIO2, however in our experiments we used identity for rotation with no translation. All data were recorded using ROS 2 bag files for both real-time and offline processing. For running the mapping system, both in real-time and during offline refinement, a separate computer was used. It featured an AMD Ryzen 7 CPU, NVIDIA RTX 3060 GPU, and 32 GB RAM, running Ubuntu 22.04 with ROS 2 Humble. All mapping modules were implemented in C++ to ensure real-time performance and efficient offline processing.

4.1.2 Dataset

To evaluate the performance of the proposed mapping system, experiments were performed with various open source dataset and custom dataset collected around Saxion buildings. The custom datasets containing four sequences were collected in different days, with varying trajectory length, and number of loop closures. Three of the sequences denoted as Saxion-00, Saxion-01, and Saxion-03 were recorded in outdoor environment while one sequence referred to as Saxion-11 was recorded in indoor. Saxion-01 is the shortest of the outdoor trajectories and contains places revisited only in the same direction. Saxion-02 includes both reverse direction and lane-level lateral shift in revisited areas. Saxion-03 contains the longest trajectory with the highest number of loop closures.

Public datasets, MulRan¹ [41] and KITTI² [42], were used to evaluate the robustness of the proposed mapping module to different variety of sensors types, trajectory length, and driving speed. These open source dataset also contain ground truth trajectories that are used for comparing the performance of the proposed method. The sequences KAIST02, KAIST03, and RiverSide01 from MulRan and sequences 00, and 05 from KITTI are used in the experimentation. Table 4.1 contains details of the datasets used in our experimentation.

¹<https://sites.google.com/view/mulran-pr/dataset>

²<https://www.cvlibs.net/datasets/kitti/>

Table 4.1: Dataset used for evaluating performance of our mapping module

Dataset	LiDAR	Envt.	G.Truth	Size	Dist.(m)	Duration(s)
Saxion-01	OS1-128	Outdoor	✗	11GB	935	740
Saxion-02	OS1-128	Outdoor	✗	29GB	1155	958
Saxion-03	OS1-128	Outdoor	✗	50GB	1823	1668
Saxion-11	OS1-128	Indoor	✗	54GB	431	696
KITTI05	HDL-64E	Outdoor	✓	21GB	2205	288
KAIST03	OS1-64	Outdoor	✓	18GB	6272	862
KITTI00	HDL-64E	Outdoor	✓	34GB	3717	470

In order for the SLAM to work correctly for each dataset, an extrinsic calibration between the IMU and LiDAR sensors is required. Table 4.2 shows the calibration matrix for each dataset used in the experiments.

Table 4.2: Extrinsic calibration between IMU and LiDAR for the different datasets we used

Dataset	Rotation	Translation(x, y, z)
Saxion	Identity	0, 0, 0
KITTI	Identity	0.81, -0.32, 0.8
KAIST	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	1.77, 0.0, -0.05

4.2 Evaluation metrics

The proposed mapping pipeline was evaluated using a comprehensive set of quantitative and qualitative metrics to ensure a robust performance assessment. Specifically, Absolute Pose Error (APE) was employed to evaluate the global consistency of the estimated trajectory by comparing it against the ground truth data. To measure local accuracy and drift between consecutive poses, Relative Pose Error (RPE) was employed. To compute the APE, and RPE values of two trajectories, the Evo [43] tool that supports trajectories in different file formats was used. All the APE values in our experiments are computed after aligning the trajectories using Evo.

Map consistency was assessed qualitatively through visual inspection of the generated maps and quantitatively using Mean Map Entropy (MME) [44]. It computes the natural

logarithm of the determinant of the covariance matrix. The smaller the MME value is, the more consistent the map is. Finally, Runtime Performance was measured by recording the average processing time per LiDAR scan in the online module. These evaluation criteria collectively provide a thorough insight into the accuracy, robustness, and efficiency of the developed system.

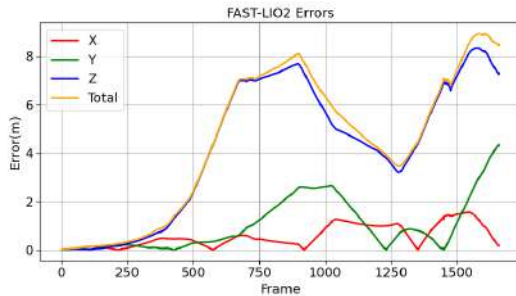
4.3 Results

This section presents the results obtained from the proposed mapping system, including maps and trajectories generated using three components; FAST-LIO2, FAST-LIO2 and PGO, and HBA for all the datasets mentioned in Table 4.1. These results are analyzed to evaluate the accuracy and consistency of the mapping performance. For this purpose, comparison of both the maps and trajectories in each stage the mapping pipeline was performed.

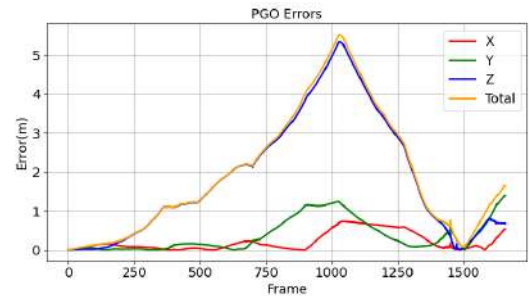
4.3.1 Trajectory accuracy

Table 4.3: Absolute Pose Error (APE) with respect to translation (in meters) for `saxion-01`.

Method	Translation Error (m)				
	Max	Mean	Median	RMSE	STD
FAST-LIO2	8.90	4.46	4.81	6.33	2.93
PGO	5.50	1.95	1.48	2.51	1.58



(a)



(b)

Figure 4.2: APE of FAST-LIO2 and PGO trajectories for `Saxion-01` dataset w.r.t HBA trajectory.

For the KITTI and MulRan datasets, trajectories of the proposed system are compared with the ground truth trajectory and then APE, and RPE are computed. Since the custom datasets do not have ground truth trajectory, the refined trajectory generated by HBA was used as ground truth and compared with FAST-LIO2 and PGO trajectories.

Table 4.4: RMSE of the APE for each Saxion dataset before and after PGO

Method	Dataset			
	saxion-01	saxion-02	saxion-03	saxion-11
FAST-LIO2	6.33 m	3.71 m	2.10 m	0.90 m
tra. (m), rot. ($^{\circ}$)	2.0 $^{\circ}$	7.35 $^{\circ}$	1.03 $^{\circ}$	0.56 $^{\circ}$
PGO	2.51 m	4.04 m	0.88 m	0.64 m
tra. (m), rot. ($^{\circ}$)	1.4 $^{\circ}$	1.93 $^{\circ}$	0.93 $^{\circ}$	0.57 $^{\circ}$

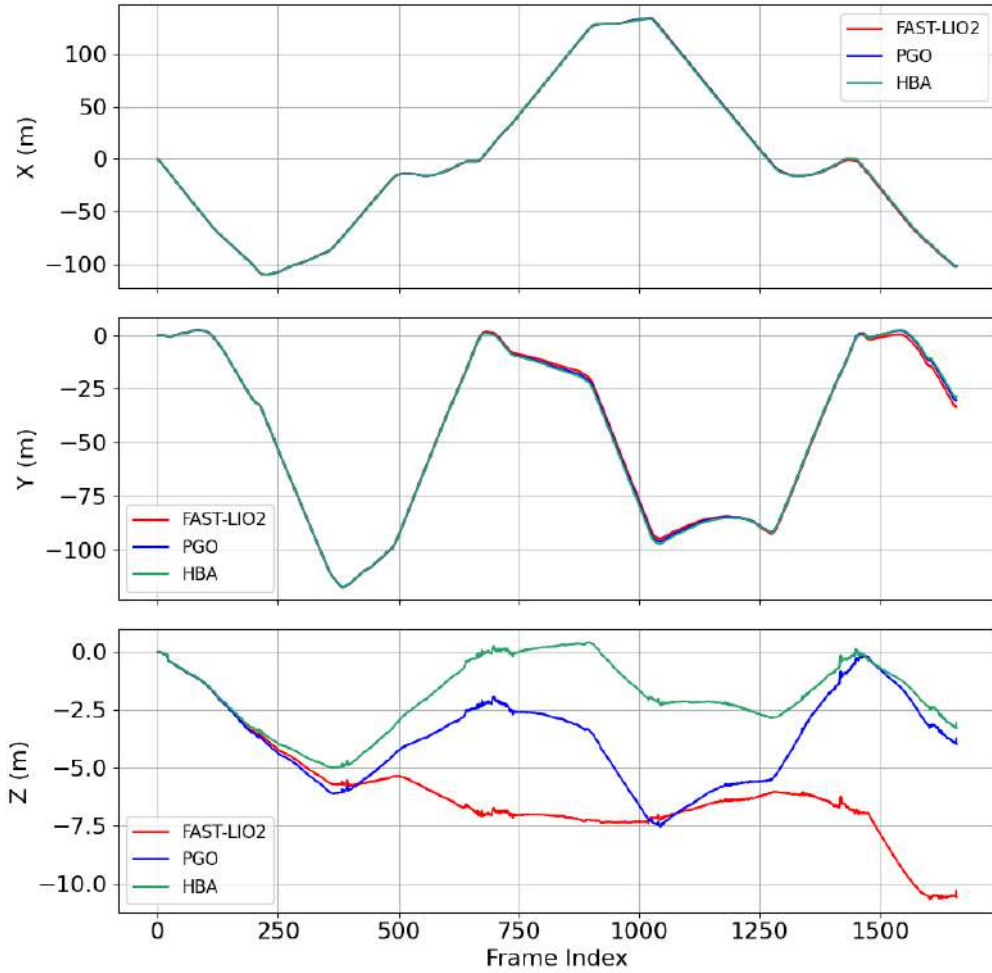


Figure 4.3: Axis-wise comparison of the trajectories of FAST-LIO2, PGO, and HBA

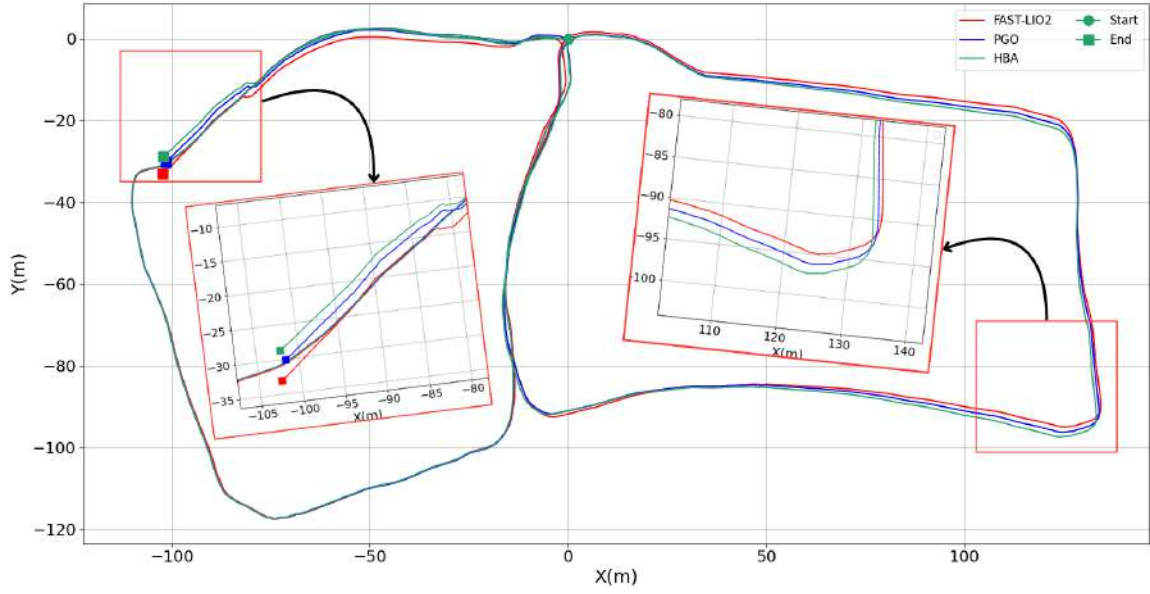


Figure 4.4: Comparison of trajectories for Saxion-01 dataset.

Figure 4.4 shows the trajectories generated by FAST-LIO, PGO, and HBA. It can be shown that the trajectory has progressively been improved by loop closure with PGO, and then by HBA. Considering the HBA trajectory as ground truth, we compute the APE with both the odometry and PGO trajectories and found that the odometry has Root Mean Squared Error (RMSE) of **6.3m** and PGO has RMSE of **2.51m**. Table 4.3 shows the APE with respect to translation part for the three outdoor Saxion datasets.

The RMSE for the estimated motion across all Saxion dataset sequences, including the indoor dataset, is presented in Table 4.4, using HBA as the reference. The errors for FAST-LIO2 and PGO are illustrated in Figure 4.2a and Figure 4.2b, respectively.

4.3.2 Map quality and consistency

The evaluation of map accuracy and global consistency was conducted by analyzing the outputs generated after each stage of the mapping pipeline for the custom and public datasets. In the initial stage, maps generated using FAST-LIO2 alone demonstrated locally smooth and accurate map, especially in short trajectories. For instance, for saxion01 dataset, the map in Figure 4.5a showed minimal distortion within the sequential sub-segments. This is due the tightly coupling of LiDAR and IMU. However cumulative pose error caused noticeable misalignment in the loop closure regions. Specifically, a lateral

misalignment of up to 9m (Table 4.3) was observed, indicating the limitation of FAST-LIO2 mapping.

Upon incorporating PGO, the system demonstrated significant correction in the loop closure areas bringing the lateral misalignment down to 2.5m, see Figure 4.5b, for saxion01 dataset. The vertical misalignment in the map (Figure 4.5c) and in the trajectory (Figure 4.3) was also significantly corrected. This vertical drift is common in most LIO methods and we incorporated GPS factor into the PGO only considering the altitude difference as a factor and putting large values for the x, and y in the covariance matrix. The map after correcting the vertical misalignment is shown in Figure 4.5d. The PGO map have small misalignment caused by inaccuracies in the loop closure and incorrect relative pose (constraints) between nodes in the pose graph.

The application of HBA to the frames (poses) generated by PGO in the offline phase yielded notable improvement in the map quality (Figure 4.5e) and trajectory (Figure 4.4). The residual distortions that persisted after PGO are corrected and refined. In the saxion01 dataset, the 2.5m lateral misalignment was eliminated, see Figure 4.5e, after applying HBA. Figure 4.6 shows the final mapping result for the saxion03 dataset.

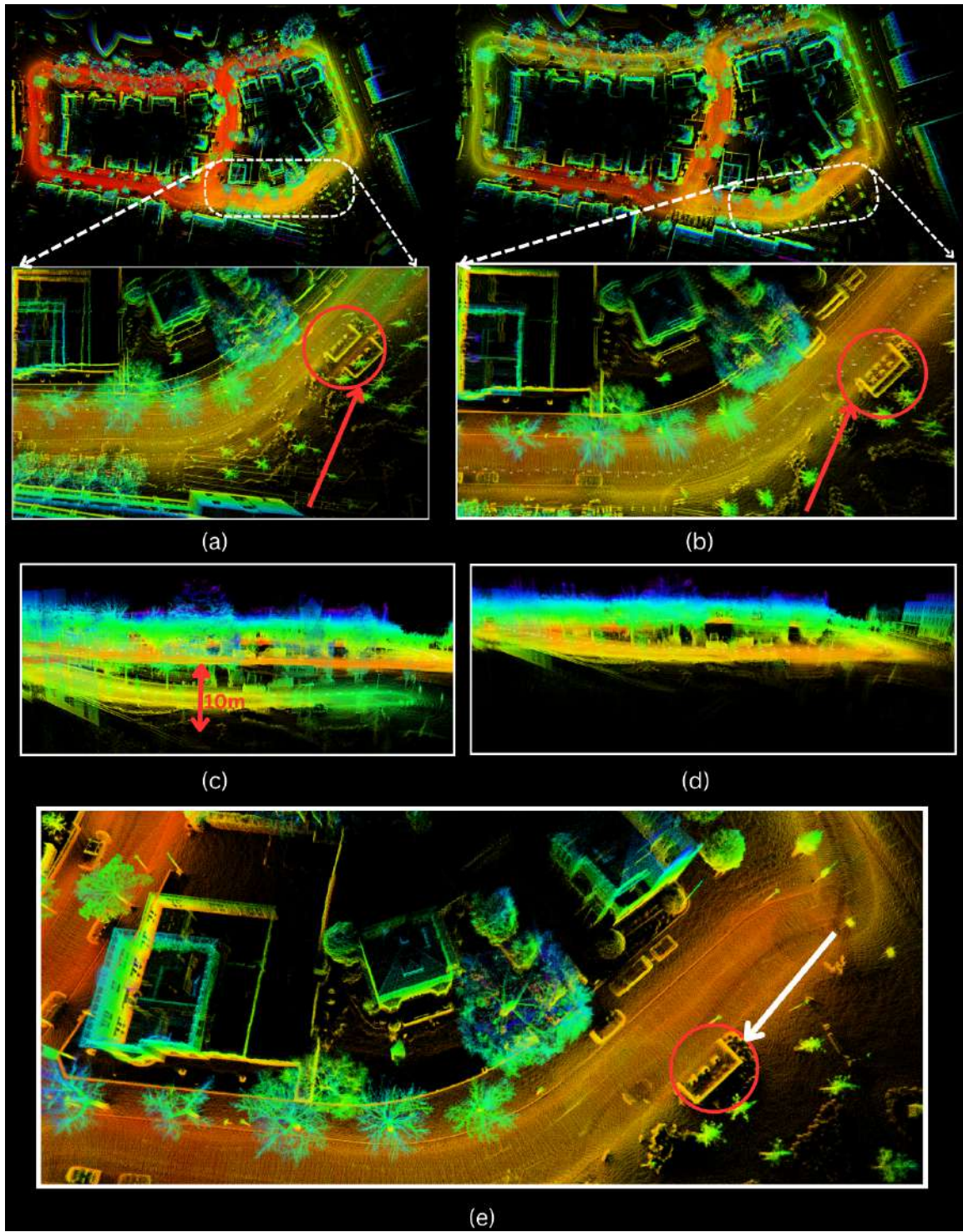


Figure 4.5: Map generated by each stage of the mapping pipeline for saxion01. (a) FAST-LIO2 top view. (b) PGO top view. (c) FAST-LIO side view. (d) PGO side view. (e) HBA

Table 4.5: $MME(\mu, \sigma)$ of map generated from the different datasets after each mapping stage. The MME values in **bold** demonstrate the best quality of the map and those in underline show the second best quality of the map.

Method	S01	S02	S03	S11	K03	K00
FAST-LIO2	-2.63, 0.11	-1.9, 0.19	-2.46, 0.19	-3.34 , 0.10	-1.85, 0.07	-2.40, 0.04
PGO	<u>-2.66</u> , 0.14	-2.50 , 0.12	<u>-2.60</u> , 0.30	<u>-3.21</u> , 0.20	<u>-1.91</u> , 0.03	<u>-2.47</u> , 0.08
HBA	-2.71 , 0.05	<u>-2.48</u> , 0.084	-2.85 , 0.13	-3.20, 0.091	-2.39 , 0.05	-2.97 , 0.026

S stands for Saxion, K stands for KAIST, $\mathcal{K}$ stands for KITTI, and

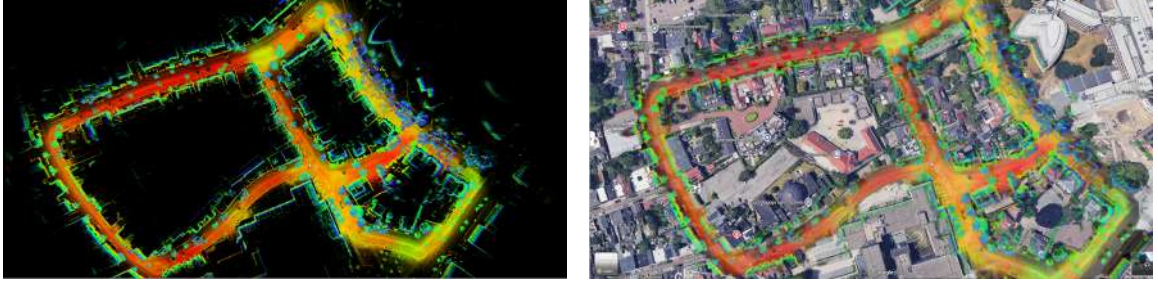


Figure 4.6: Mapping result for `saxion03`

The results clearly demonstrate the incremental improvement provided by each module, FAST-LIO2 for real-time local accuracy, PGO for global loop correction, and HBA for the full map refinement resulting in high-fidelity and metrically consistent maps. Table 4.5 shows the MME of the map generated after each stage of the mapping pipeline for the different datasets including the public ones.

4.3.3 Runtime performance

The three components of the entire mapping system are evaluated for runtime performance based on their time consumption. FAST-LIO2 runs at a frequency that is equivalent to the sensor frame rate and outputs odometry at 10HZ. The total processing time per scan includes the time taken by motion compensation, state estimation, KNN search, and mapping. To further reduce the time consumption, we disabled the mapping step and only used the it to output odometry. Total of 16 cores were used to search for the closest surface and residual computation in the IEKF step. This helped to speed up the process by utilizing multi-threading. The average per scan processing time for FAST-LIO2 was therefore $2.5ms$.

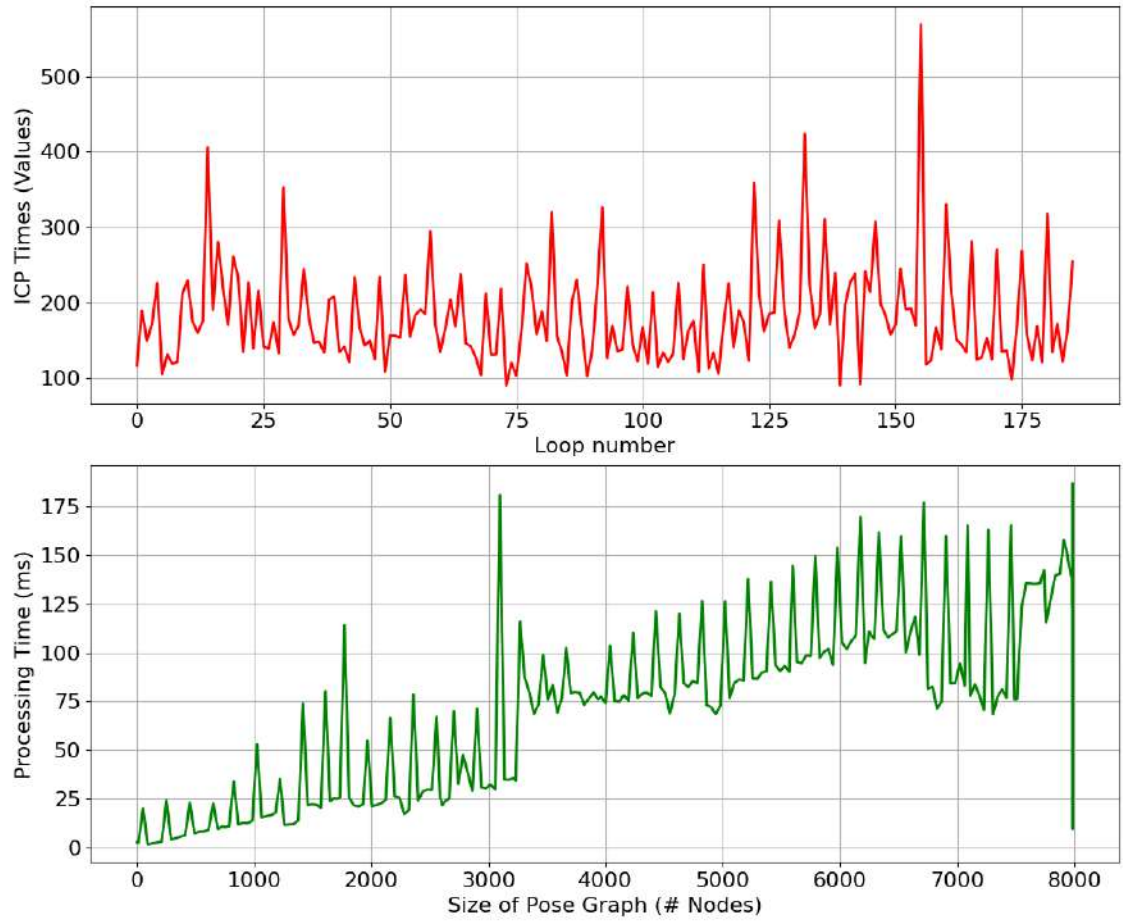


Figure 4.7: Processing times of PGO module

The total processing time for the PGO module includes loop detection, ICP alignment, pose graph construction, and solving the least-squares problem. We evaluated the processing time of the PGO module across all datasets used, and in Figure 4.7, we present the evaluation results for a dataset with a moderately long trajectory. Majority of the processing time—approximately 200 ms on average—is consumed by the ICP alignment step. The time required to optimize the pose graph using iSAM2, increases linearly with the number of nodes in the graph. For instance, processing a pose graph with up to 8000 nodes takes around 150 ms on average. To enhance the efficiency of the PGO module, we divided the workflow into subtasks—loop detection, ICP alignment, pose graph construction, and optimization—and executed each subtask concurrently in a dedicated thread.

The total computation time for HBA includes the time taken for adaptive-voxe construction, and the time taken for BA. It is also important to note that the computation

time and RAM consumption of HBA also depends on the number of poses N and the optimal number of layers l . In complex environments, map construction takes longer due to the increased computational cost of adaptive voxel map generation. If the number of frame poses is larger ($N > 5 \times 10^3$), a 4-layer pyramid is optimal, and for $N < 5 \times 10^3$, 3-layer and 4-layer pyramid perform similar[30].

4.4 Comparison with state of the art

Table 4.6: MME comparison with state of the art.

Method	S01	S02	S03	K05	K03	K00
K-SLAM	<u>-2.47</u> , 0.05	<u>-2.53</u> , 0.08	-2.52, 0.11	-2.54, 0.16	-2.42, 0.03	-2.50, 0.13
MOLA	-2.43, 0.09	-2.54 , 0.05	-2.67, 0.045	-2.66 , 0.05	-1.79, 0.07	-2.63 , 0.045
GLIM	-2.40, 0.12	-2.49, 0.09	<u>-2.73</u> , 0.15	-2.49, 0.05	-2.35, 0.044	-1.10, 0.17
Proposed	-2.71 , 0.05	-2.47, 0.084	-2.85 , 0.13	<u>-2.56</u> , 0.066	<u>-2.39</u> , 0.05	<u>-2.57</u> , 0.026

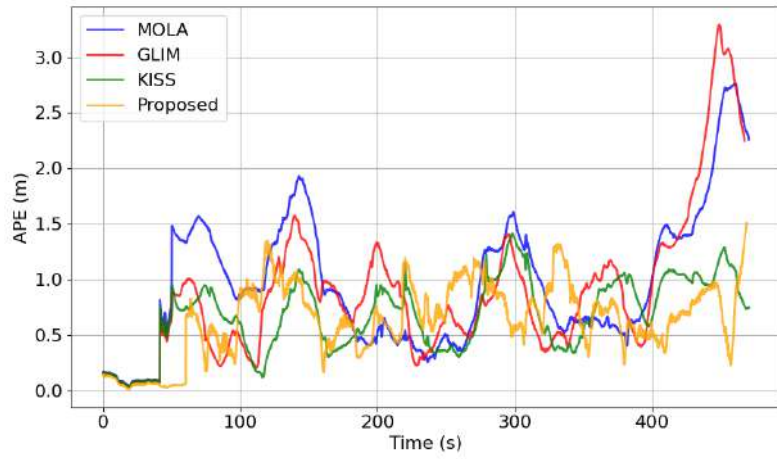
Values in **bold** and underline show the best and 2nd best quality of the map respectively.

This section presents a comparative evaluation of the proposed SLAM system with three state-of-the-art methods: 3D Range-Inertial Localization and Mapping with GPU-Accelerated Scan Matching Factors (GLIM [45]), KISS SLAM[26], and MOLA SLAM[25]. The comparison is based on two criteria: map quality and trajectory accuracy, measured using MME and APE, respectively. To better evaluate the robustness of our method, the comparison was performed on the three custom datasets and three public datasets with relatively longer trajectory. Since there is no ground truth for our custom datasets, only MME evaluation on the maps was performed. For the public datasets, performance was evaluated using both MME and APE.

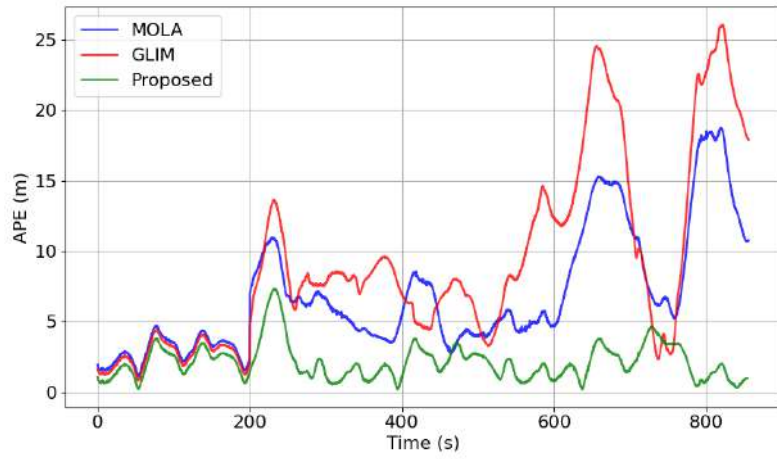
Table 4.7: Comparison of RMSE of the APE(translation) for the public dataset using our method and other SLAM methods. The values for the method that resulted in smallest error is shown in bold.

Dataset	SLAM method			
	KISS	MOLA	GLIM	Proposed
K05	2.20 m	1.80 m	2.10 m	2.80 m
K03	36 m	9.95 m	15.0 m	2.42 m
K00	3.06 m	4.04 m	3.78 m	2.55 m

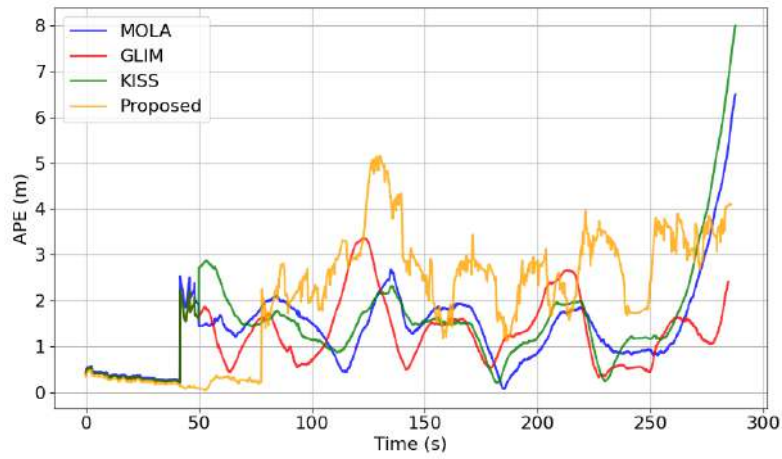
K stands for KAIST, and $\mathcal{K}$ stands for KITTI



(a) Errors on KITTI00



(b) Errors on KAIST03



(c) Errors on KITTI05

Figure 4.8: Trajectory Error comparison with other SLAM methods.

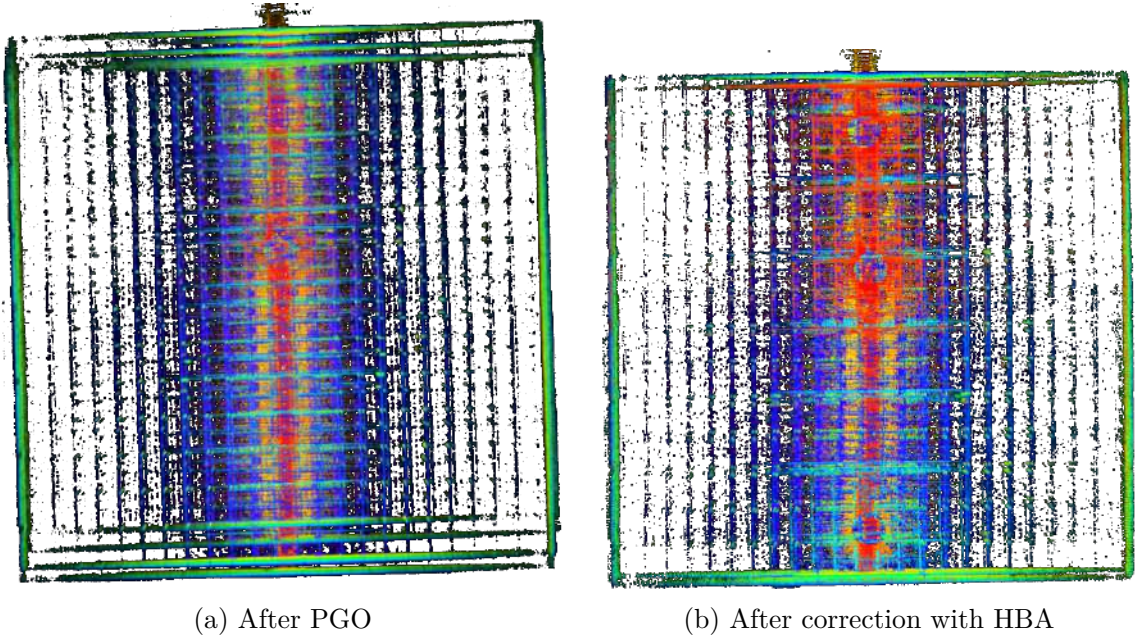


Figure 4.9: Greenhouse map

During map comparison using MME, the final map for MOLA SLAM (saved in a metric map(.mm) format) is first converted to .pcd format before calculating MME. For GLIM SLAM and KISS SLAM, both frame poses and corresponding pcd files are outputted. Hence, before calculating MME, map was constructed by aggregating the point cloud files using frame poses. Table 4.6 shows the MME of the maps generated for the custom and public datasets using the different SLAM methods including the proposed. ATE was computed only for KITTI00, KITTI05, and KAIST03 as shown in Table 4.7 because these datasets have ground truth trajectories.

Chapter 5

Discussion

This chapter presents interpretation of the results obtained from the experiments, highlighting the strengths and limitations of the proposed hybrid LiDAR SLAM framework. The discussion connects the findings with the original research objectives, evaluates their implications in both indoor and outdoor scenarios, and reflects on how the integration of FAST-LIO2, loop closure with Scan Context and factor graph, and bundle adjustment contributed to global consistency. Furthermore, it explores the challenges encountered during implementation and analysis, providing insights that can inform future enhancements and deployments of SLAM systems in complex environments.

5.1 Interpreting the results

The experimental results demonstrate the effectiveness of the proposed hybrid LiDAR SLAM pipeline in achieving globally consistent mapping across diverse indoor and outdoor environments. The trajectory accuracy and map consistency metrics reveal that integrating FAST-LIO2 with PGO significantly reduces accumulated drift compared to using FAST-LIO2. Moreover, applying HBA on top of PGO optimized map further refines the global consistency of the map.

5.1.1 Trajectory and Map Accuracy

As shown in Table 4.4, the APE of the translation for **Saxion-01** decreases from 6.33 m with FAST-LIO2 to 2.51 m after applying PGO. Figure 4.5 illustrates this correction, where a vertical misalignment of about 10 m in the revisited area (Figure 4.4c) is corrected after PGO (Figure 4.5d). However, residual misalignments remain after PGO (Figure 4.5b

and Figure A.2e), due to ICP inaccuracies and the fact that PGO does not globally optimize point cloud consistency. Applying HBA further refines the map, resulting in a more consistent reconstruction (Figure 4.5e).

For **Saxion-02**, the APE increases slightly from 3.71 m to 4.04 m due to limited loop closures and a 15 m lateral shift in the revisited area, which caused loop detection to fail. Although radius-based search was used to compensate for Scan Context’s failure, ICP introduced alignment errors. In contrast, **Saxion-03** has multiple revisited areas, and the use of PGO and HBA significantly improved both trajectory accuracy and map quality.

In **Saxion-11**, FAST-LIO2 alone performs well with minimal drift, as the structured indoor environment (Figure A.6) contains abundant geometric features like edges and planes, providing strong constraints for scan registration.

The MME scores in Table 4.5 confirm that PGO and HBA significantly improve map quality across all datasets by enhancing global consistency. The only exception is the indoor **Saxion-11**, where FAST-LIO2 already produces a consistent map (Figure A.6) due to the structured environment, leaving little room for further improvement.

5.1.2 Discussing the ATE and MME comparison results

Table 4.7 presents the RMSE of the trajectories of the proposed method and other SOTA SLAM methods for the three public datasets. The RMSE of the pair of trajectories is computed after they are aligned using Umeyama algorithm in Evo. The best-performing method for each dataset is highlighted in bold. For the KITTI05 dataset, MOLA achieves the lowest error at **1.80m** and the proposed method achieves **2.80m**. For KAIST03 dataset, the proposed method outperforms the other methods with an error of **2.42m**, significantly lower than the other methods. It is important to note that KISS SLAM results were not aligned due to missing timestamps in its outputs, leading to high errors and an overly optimistic comparison. Similarly, for the KITTI00 dataset, the proposed method performs best with RMSE of **2.55m**.

Figure 4.8 provides a detailed visual comparison of the APE, complementing the RMSE summary in Table 4.7. The Figures 4.8a, 4.8b, and 4.8c shows the APE over time computed for the entire trajectory of KITTI00, KAIST03, and KITTI05 respectively.

Overall, proposed method demonstrated better performance indicating its potential for reliable pose estimation across different scenarios.

Table 4.6 presents the MME comparison of the proposed method and other SLAM methods across the datasets. The proposed method consistently achieves either the best or second-best results in all sequences, indicating superior or comparable map quality. Notably, it outperforms other methods in five out of seven cases, demonstrating its effectiveness in generating accurate and consistent map both for indoor and outdoor environments.

5.2 Challenges

One of the primary challenges faced was the failure in loop closure detection when a place is revisited in a reverse direction or there is a lane-level lateral shift in revisiting a place, see Figure 5.1. This issue was exacerbated when the horizontal FoV of the LiDAR is partially occluded or unavailable, leading to a complete failure in loop detection. As a fallback, a radius based search was employed to identify potential loop candidates from which the one with smallest registration error was chosen as a loop. However, subsequent ICP registration often yielded inaccurate alignment and even sometimes introducing false positives due to inefficiency of the radius-based search.

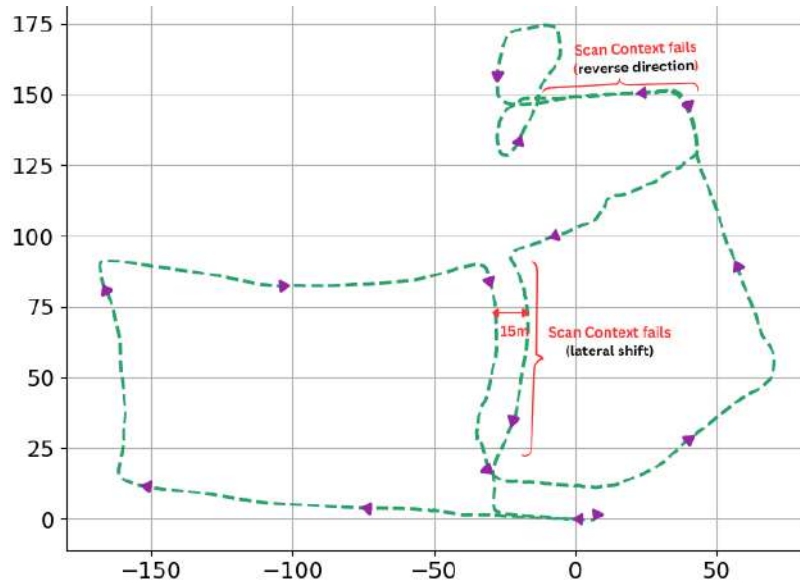


Figure 5.1: Loop detection failure case in lane-level later shift and reverse revisit

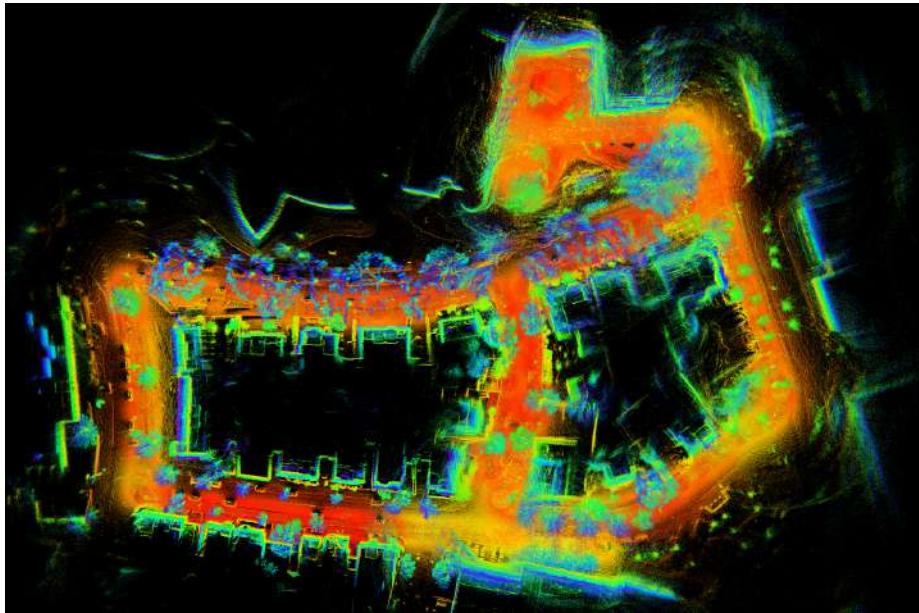


Figure 5.2

Figure 5.3: Effect of loop detection failure in the map

Another critical challenge was the occurrence of false positive loop closures in environments with repetitive structures such as greenhouses, and bridges presented similar features over long stretches. This makes it difficult for the loop detection algorithm to distinguish between truly revisited locations and visually similar but distinct areas. These false matches led to incorrect loop closures, compromising the overall map consistency. Figure 4.9a shows the effect of such false loop closures in consistency of the map.

Chapter 6

Conclusion

This chapter summarizes the key findings of the research, revisits the research questions relating to the results, highlights the main contributions of the work, and offers recommendations for future research. The study aimed to develop a robust LiDAR-based SLAM framework that integrates LiDAR-inertial odometry, loop closure, and global optimization to generate globally consistent maps in diverse environments. Through extensive experiments across indoor and outdoor datasets, the system's performance has been evaluated, demonstrating its effectiveness in addressing the challenges of real-time mapping and global map consistency in SLAM.

6.1 Summary of Findings

The proposed SLAM system, demonstrated significant improvements in trajectory accuracy and map consistency across multiple datasets. Quantitative results showed that PGO effectively reduced accumulated drift in real-time but does not optimize the map consistency, while HBA further refined global alignment and improved map consistency. The system performed particularly well in indoor and large-scale outdoor environments with frequent loop closures. Overall, the hybrid approach successfully addressed the limitations of individual modules, yielding a robust and consistent mapping pipeline.

6.2 Research questions revisited

1. Is LiDAR Inertial Odometry sufficient to generate accurate map of complex environments?

LiDAR Inertial Odometry frameworks provide accurate short-term motion estimates. FAST-LIO2, for instance, generates accurate and consistent map for short trajectories, especially for indoor environments with more structured features. However, in large-scale or long-duration scenarios, it accumulates drifts and fails to maintain global map consistency.

2. What are the limitations of Graph SLAM in generating consistent map especially in revisited locations?

Graph SLAM, while effective at correcting drift through loop closures, relies heavily on accurate loop detection and alignment. When loop closures are missed or poorly aligned, it can lead to residual errors or duplicated structures in the map. Moreover, PGO alone does not enforce point cloud consistency as it only optimizes the relative pose constraints between nodes in the graph.

3. How effective is it to integrate LiDAR Inertial Odometry, Graph SLAM, and LiDAR Bundle Adjustment for generating globally consistent map of environments?

The integration of LiDAR inertial odometry, PGO, and HBA proved highly effective in improving trajectory accuracy and map consistency. FAST-LIO2 offers reliable real-time odometry, while PGO reduces drift through loop closure. HBA further refines the global structure by jointly optimizing poses and point clouds. While FAST-LIO2 and PGO are able to run in real-time, due to its computational cost, HBA is performed offline and it significantly enhances map quality in complex or large-scale environments where global accuracy is essential.

6.3 Contributions

The key contribution of this work is the integration of PGO as an effective bridge between FAST-LIO2 and HBA. While FAST-LIO2 provides accurate local motion estimation, it accumulates significant drift over long trajectories, which results in inconsistent global maps. Directly applying HBA to FAST-LIO2 output is ineffective in such cases, as HBA

is limited to correcting divergences smaller than the voxel size used during optimization and cannot resolve large-scale drift. By introducing PGO between FAST-LIO2 and HBA, the system first reduces drift through loop closure constraints, producing an initial globally consistent trajectory, in real time, that falls within the convergence range of HBA. This makes HBA effective in further refining both trajectory and point cloud alignment. Moreover, PGO addresses challenges in revisited areas and helps mitigate duplication in the map, which HBA alone cannot resolve. This structured integration of local odometry, loop closure-based correction, and global refinement leads to a SLAM system that is both more reliable and capable of producing consistent maps across complex environments.

6.4 Future work

Although the proposed hybrid LiDAR SLAM system demonstrates promising results in generating globally consistent maps for both indoor and outdoor environments, several directions can be explored to further improve the system:

- **Robust Loop Closure with Deep Learning:**

Incorporating deep learning-based loop closure methods can improve detection accuracy in perceptually challenging environments. Learned descriptors can better capture semantic and geometric features, enabling more reliable loop detection across varying viewpoints and conditions.

- **Dynamic Object Removal:**

Filtering out dynamic objects such as pedestrians and vehicles can significantly improve map consistency. Future work could use motion segmentation, temporal filtering, or semantic classification to exclude non-static elements from the mapping process.

- **Camera-LiDAR Fusion:**

Combining visual data with LiDAR can enhance performance in environments with sparse geometric features. The fusion of semantic-rich camera input with LiDAR geometry can improve localization, mapping accuracy, and overall scene understanding.

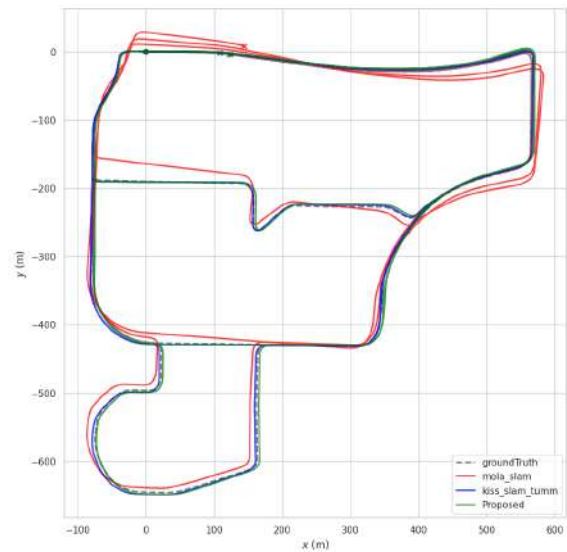
Appendix A

Appendices

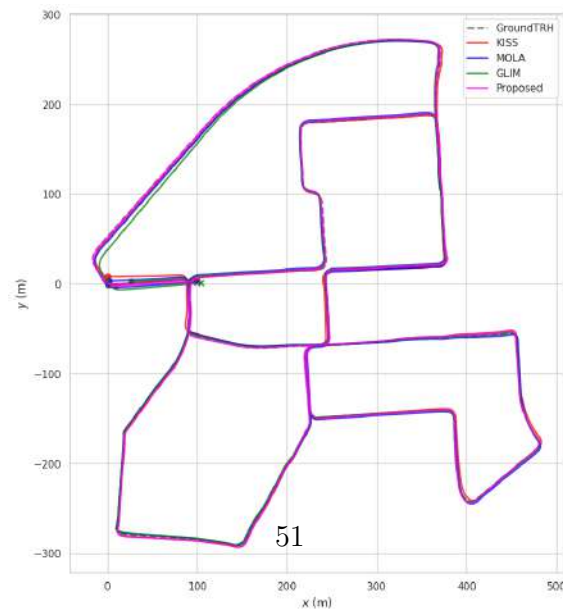
A.1 Trajectories of public datasets



(a)



(b)



A.2 Results for MulRan KAIST3 sequence

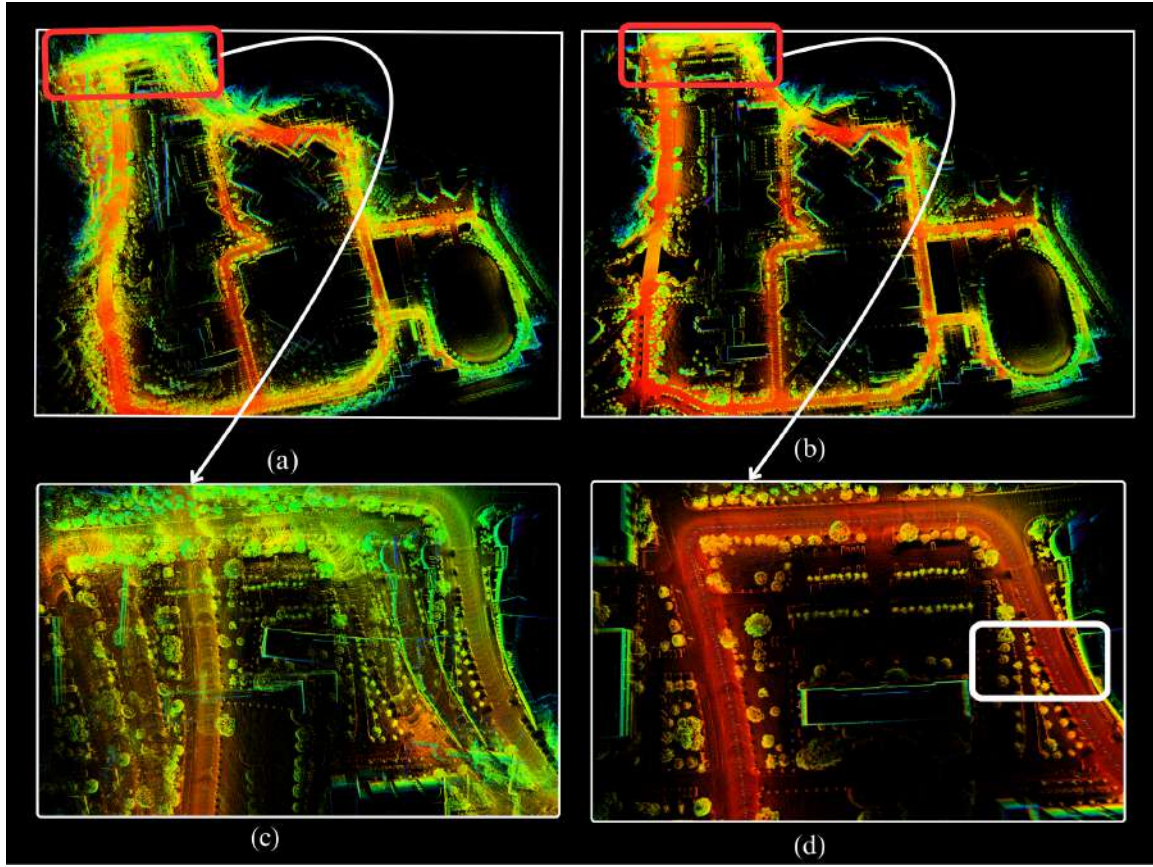


Figure A.2: Map generated by each stage of the mapping pipeline for MulRan KAIST dataset sequence 03. (a) Map generated by FAST-LIO2, (b) Map after optimizing with PGO, (c) and (d) show segments from (a) and (b) zoomed in respectively.

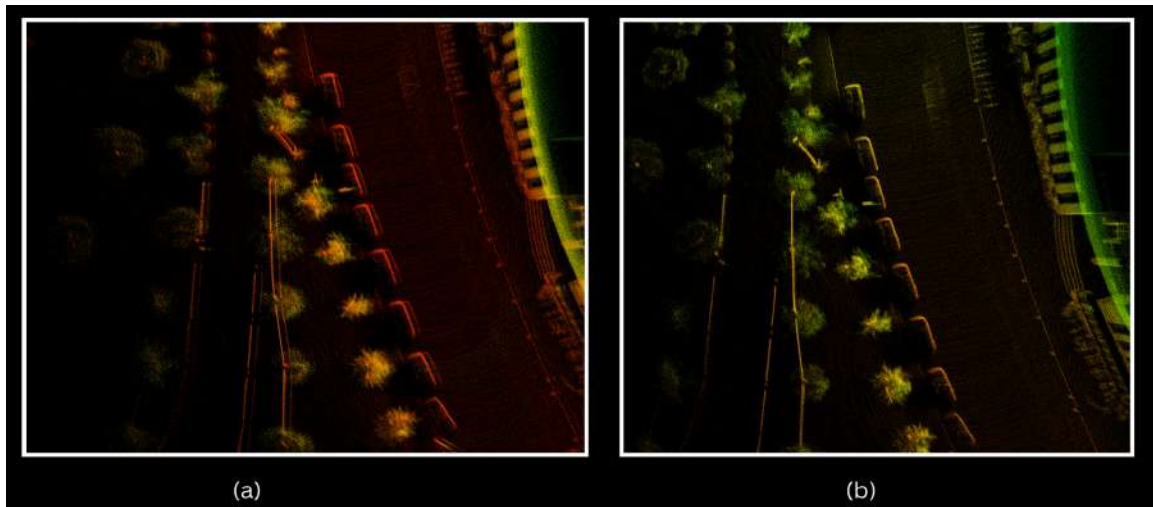


Figure A.3: (a) The zoomed in segment of map in Figure A.2d, (b) After refinement with HBA

A.3 Results for KITTI sequence 00

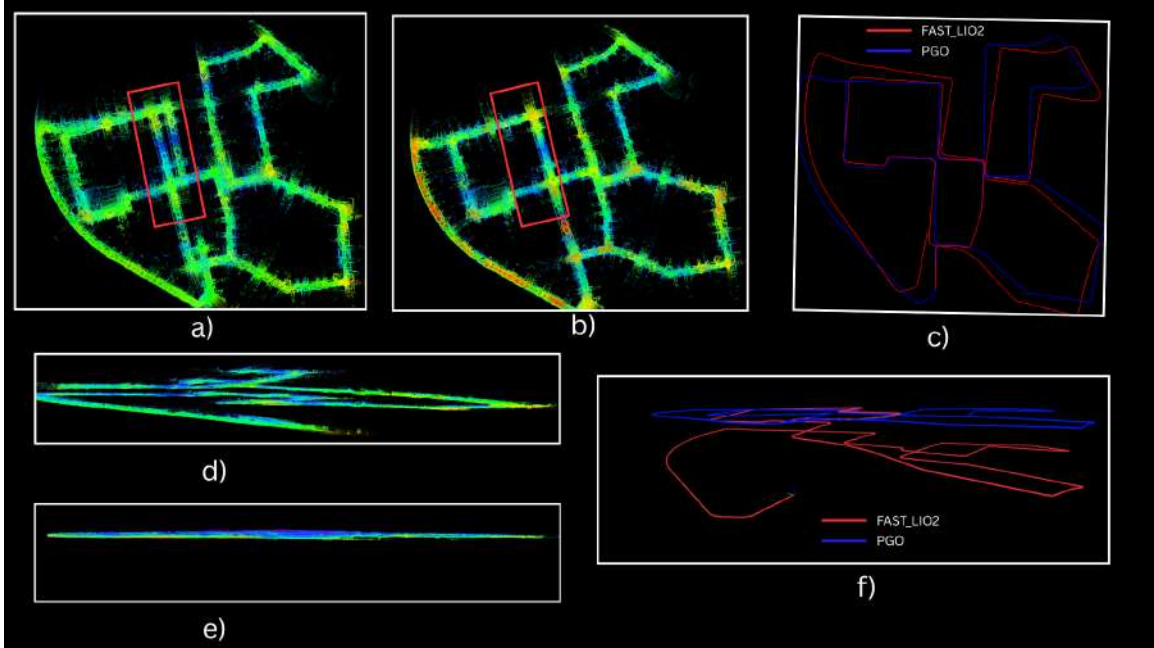


Figure A.4: Map visualization for KITTI00 sequence. (a) BEV of map generated by FAST-LIO2. (b) BEV of map after PGO optimization. (c) BEV of FAST-LIO2 and PGO trajectories. (d) Side view of FAST-LIO2 map. (e) Side view the optimized map. (f) Side view of FAST-LIO2 and PGO trajectories

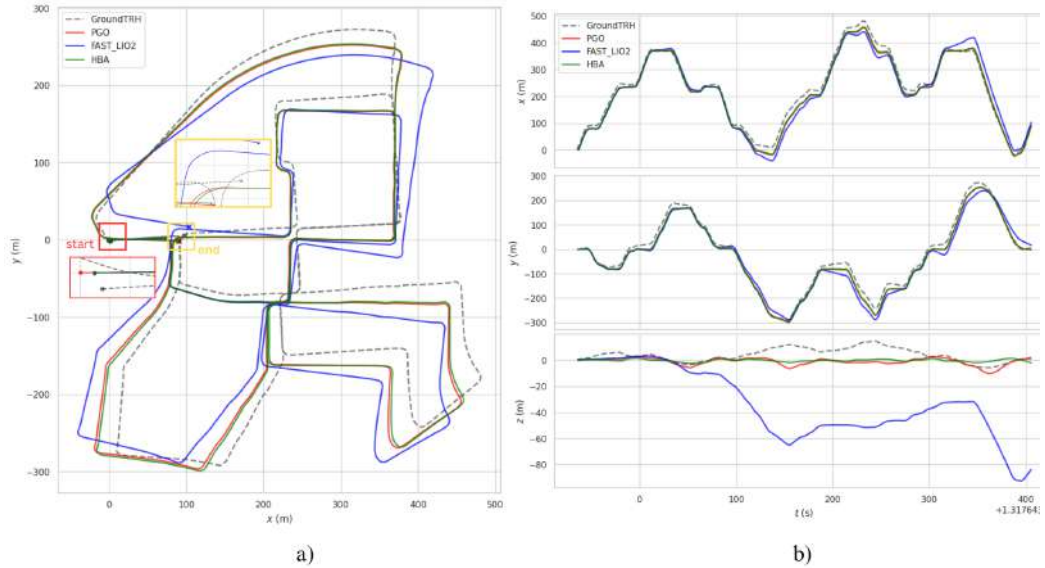


Figure A.5: Visualization of trajectories for KITTI00. a) BEV trajectories. b) Axis-wise comparison of the trajectories.

A.4 Results for Saxion11(Indoor)

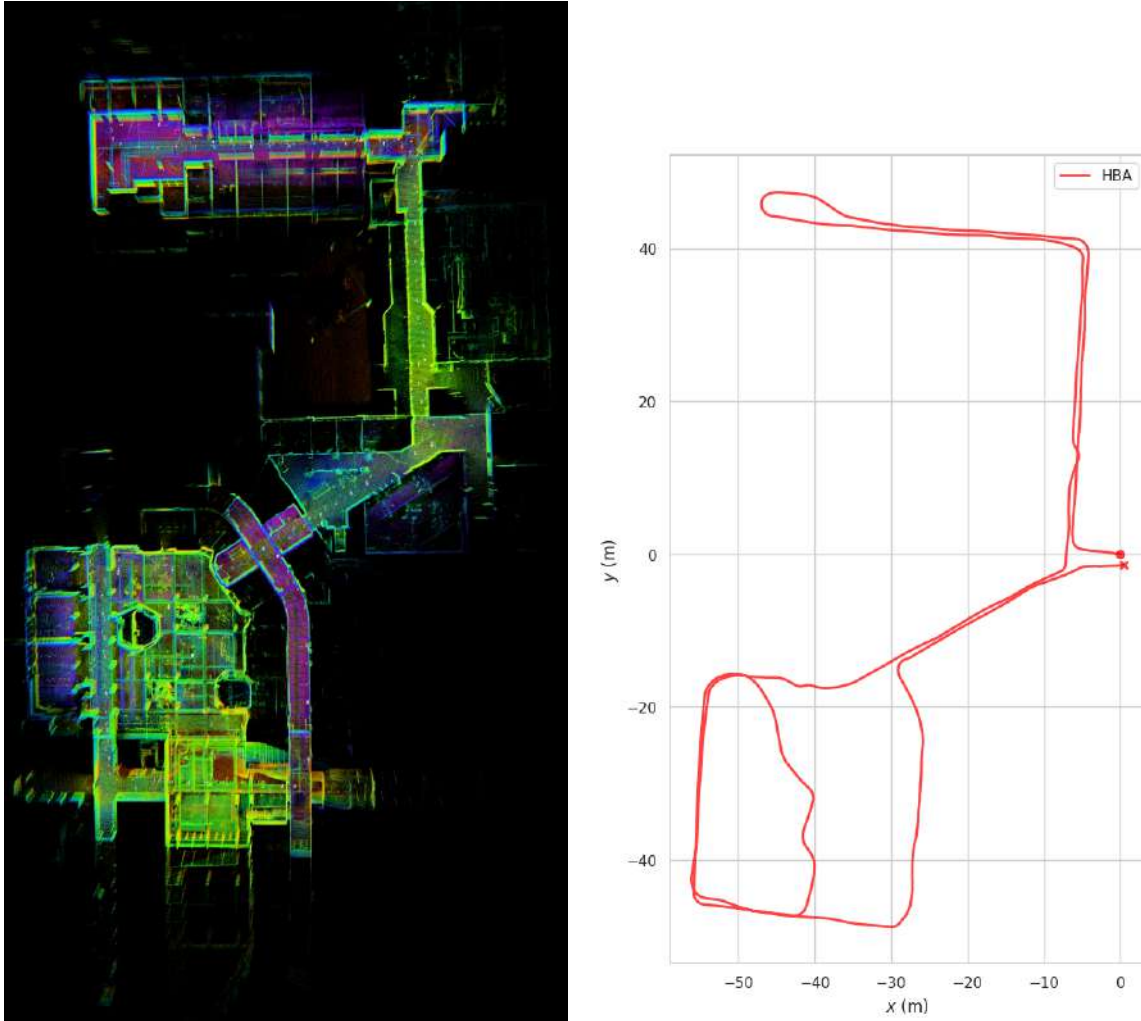
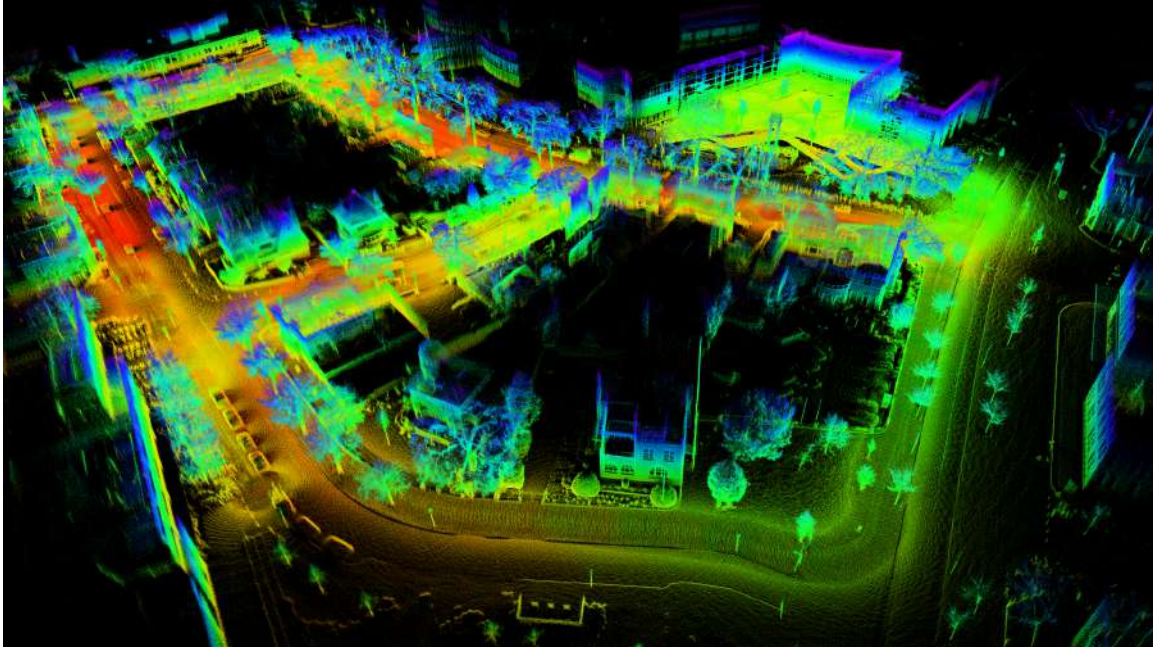
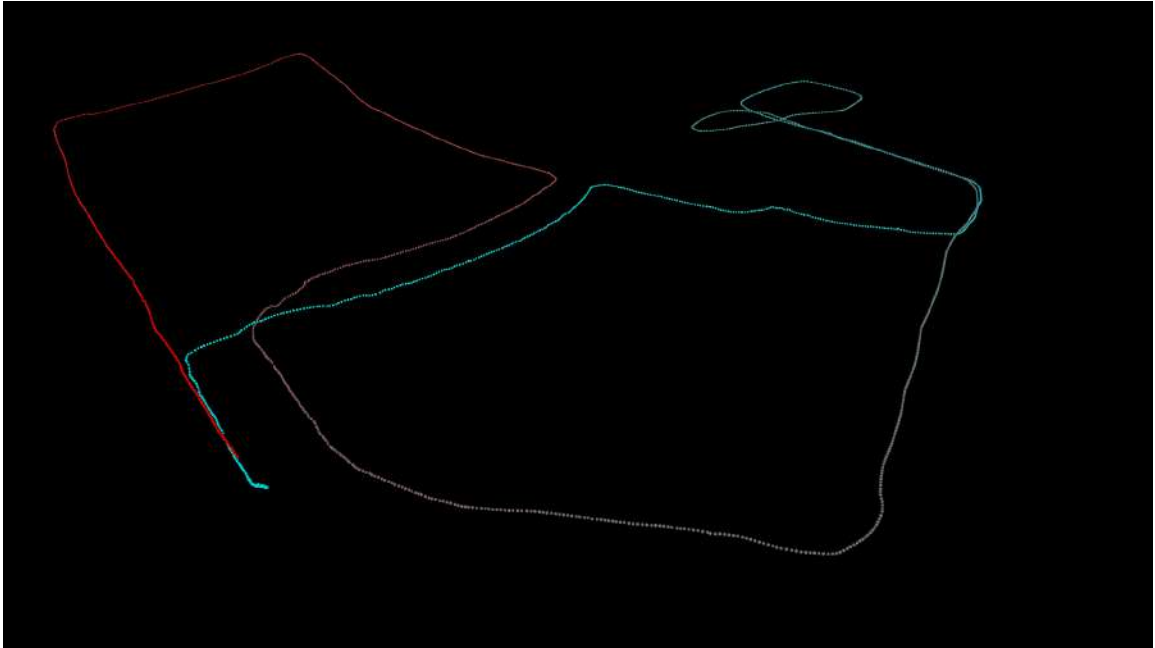


Figure A.6: Mapping result for `saxion11`[Indoor]

A.5 Results for Saxion02



(a) 3D Map



(b) Trajectory, colors evolve from start(red) to finish(cyan)

Figure A.7: Mapping result for Saxion02 after refinement with HBA

Bibliography

- [1] Alonzo Kelly. *Mobile Robotics: Mathematics, Models, and Methods*. Cambridge: Cambridge University Press, 2013, pp. 579–639.
- [2] Cesar Cadena et al. “Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age”. In: *arXiv preprint arXiv:1606.05830* (2016).
- [3] Bharath Garigipati, Nataliya Strokina, and Reza Ghabcheloo. “Evaluation and comparison of eight popular Lidar and Visual SLAM algorithms”. In: *arXiv preprint arXiv:2208.02063* (2022). URL: <https://arxiv.org/abs/2208.02063>.
- [4] Ji Zhang and Sanjiv Singh. “Low-drift and real-time lidar odometry and mapping”. In: *Autonomous robots* 41 (2017), pp. 401–416.
- [5] Rinf.tech. *Your Definitive Guide to LiDAR Technology*. <https://www.rinf.tech/your-definitive-guide-to-lidar-technology/>. Accessed: 2025-05-16. 2024.
- [6] Wen Xu et al. “FAST-LIO: A Fast, Robust LiDAR-inertial Odometry Package by Tightly-coupled Iterated Kalman Filter”. In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 7270–7276.
- [7] Yongjun Zhang, Pengcheng Shi, and Jiayuan Li. “3D LiDAR SLAM: A survey”. In: *The Photogrammetric Record* 39.186 (2024), pp. 457–517. DOI: 10.1111/phor.12497.
- [8] Richard C Smith and Peter Cheeseman. “On the representation and estimation of spatial uncertainty”. In: *The international journal of robotics research* 5.4 (1986), pp. 56–68.
- [9] Gamini Dissanayake et al. “A solution to the simultaneous localization and map building (SLAM) problem”. In: *IEEE Transactions on Robotics and Automation* 17.3 (2001), pp. 229–241.

- [10] Rainer Kuemmerle et al. “g2o: A general framework for graph optimization”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3607–3613.
- [11] Rainer Kummerle et al. “Large scale graph-based SLAM using aerial images as prior information”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 1724–1729.
- [12] Sebastian Thrun. “Probabilistic robotics”. In: *Communications of the ACM* 45.3 (2002), pp. 52–57.
- [13] Wei Xu et al. “Fast-lío2: Fast direct lidar-inertial odometry”. In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2053–2073.
- [14] Tixiao Shan et al. “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping”. In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5135–5142.
- [15] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. “A micro lie theory for state estimation in robotics”. In: *arXiv preprint arXiv:1812.01537* (2018).
- [16] Bill Triggs et al. “Bundle adjustment—a modern synthesis”. In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.
- [17] Davide Scaramuzza and Friedrich Fraundorfer. “Visual odometry [tutorial]”. In: *IEEE robotics & automation magazine* 18.4 (2011), pp. 80–92.
- [18] Jiarong Lin and Fu Zhang. “Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV”. In: *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2020, pp. 3126–3131.
- [19] Zheng Liu and Fu Zhang. “Balm: Bundle adjustment for lidar mapping”. In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3184–3191.
- [20] Xiyuan Liu et al. “Large-Scale LiDAR Consistent Mapping Using Hierarchical LiDAR Bundle Adjustment”. In: *IEEE Robotics and Automation Letters* 8.3 (2023), pp. 1523–1530. DOI: 10.1109/LRA.2023.3238902.
- [21] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. “Analysis and improvement of the consistency of extended Kalman filter based SLAM”. In: *2008 IEEE International Conference on Robotics and Automation*. IEEE. 2008, pp. 473–479.

- [22] Frank Dellaert, Michael Kaess, et al. “Factor graphs for robot perception”. In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.
- [23] Andrea Censi. “An accurate closed-form estimate of ICP’s covariance”. In: *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE. 2007, pp. 3167–3172.
- [24] Anders Glent Buch, Dirk Kraft, et al. “Prediction of ICP pose uncertainties using Monte Carlo simulation with synthetic depth images”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4640–4647.
- [25] Jose Luis Blanco-Claraco. “A flexible framework for accurate LiDAR odometry, map manipulation, and localization”. In: *The International Journal of Robotics Research* 0.0 (2025), p. 02783649251316881. DOI: 10.1177/02783649251316881. eprint: <https://doi.org/10.1177/02783649251316881>. URL: <https://doi.org/10.1177/02783649251316881>.
- [26] Tiziano Guadagnino et al. “KISS-SLAM: A Simple, Robust, and Accurate 3D LiDAR SLAM System With Enhanced Generalization Capabilities”. In: *arXiv preprint arXiv:2503.12660* (2025).
- [27] Ji Zhang, Sanjiv Singh, et al. “LOAM: Lidar odometry and mapping in real-time.” In: *Robotics: Science and systems*. Vol. 2. Berkeley, CA. 2014, pp. 1–9.
- [28] S. Gupta et al. “Effectively Detecting Loop Closures using Point Cloud Density Maps”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024.
- [29] Giseop Kim and Ayoung Kim. “Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4802–4809.
- [30] Marian Himstedt et al. “Large scale place recognition in 2D LIDAR scans using geometrical landmark relations”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 5030–5035.
- [31] Naveed Muhammad and Simon Lacroix. “Loop closure detection using small-sized signatures from 3D LIDAR data”. In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*. IEEE. 2011, pp. 333–338.

- [32] Walter Wohlking and Markus Vincze. “Ensemble of shape functions for 3D object classification”. In: *2011 IEEE international conference on robotics and biomimetics*. IEEE. 2011, pp. 2987–2992.
- [33] Frank Dellaert and GTSAM Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. DOI: 10.5281/zenodo.5794541. URL: <https://github.com/borglab/gtsam>.
- [34] K. S. Arun, T. S. Huang, and S. D. Blostein. “Least-Squares Fitting of Two 3-D Point Sets”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.5 (1987), pp. 698–700. DOI: 10.1109/TPAMI.1987.4767965.
- [35] Gene H Golub and Christian Reinsch. “Singular value decomposition and least squares solutions”. In: *Handbook for automatic computation: volume II: linear algebra*. Springer, 1971, pp. 134–151.
- [36] Giorgio Grisetti et al. “A tutorial on graph-based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.
- [37] Michael Kaess et al. “iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3281–3288. DOI: 10.1109/ICRA.2011.5979641.
- [38] Yong Wang. “Gauss-newton method”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 4.4 (2012), pp. 415–420.
- [39] Sam Roweis. “Levenberg-marquardt optimization”. In: *Notes, University Of Toronto* 52 (1996), p. 1027.
- [40] inc. Ouster. *Ouster sensors docs*. https://static.ouster.dev/sensor-docs/image_route1/image_route3/sensor_data/sensor-data.html. Accessed: 2025-05-02. 2024.
- [41] Giseop Kim et al. “MulRan: Multimodal Range Dataset for Urban Place Recognition”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Paris, May 2020.
- [42] Andreas Geiger et al. “Vision meets Robotics: The KITTI Dataset”. In: *International Journal of Robotics Research (IJRR)* (2013).
- [43] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM*. <https://github.com/MichaelGrupp/evo>. 2017.

- [44] Jan Razlaw et al. “Evaluation of registration methods for sparse 3D laser scans”. In: *2015 European Conference on Mobile Robots (ECMR)*. 2015, pp. 1–7. DOI: 10.1109/ECMR.2015.7324196.
- [45] Kenji Koide et al. “Glim: 3d range-inertial localization and mapping with gpu-accelerated scan matching factors”. In: *Robotics and Autonomous Systems* 179 (2024), p. 104750.

Acronyms

APE Absolute Pose Error.

EKF Extended Kalman Filter.

FoV Field of View.

GNSS Global Navigation Satellite System.

GPS Global Positioning System.

HBA Hierarchical Bundle Adjustment.

ICP Iterative Closest Point.

IEKF Iterated Extended Kalman Filter.

IMU Inertial Measurement Unit.

INS Inertial Navigation System.

KF Kalman Filter.

LBA LiDAR Bundle Adjustment.

LiDAR Light Detection and Ranging.

LIO LiDAR-Inertial Odometry.

LO LiDAR Odometry.

MME Mean Map Entropy.

PDF Probability Density Function.

PF Particle Filter.

PGO Pose Graph Optimization.

RADAR Radio Detection and Ranging.

SfM Structure from Motion.

SLAM Simultaneous Localization and Mapping.

SVD Singular Value Decomposition.

UKF Unscented Kalman Filter.

17

Joint Underwater Mapping with Acoustic and Optical Images
Precious Philip-Ifabiyi

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND
COMPUTING

MASTER'S THESIS No. 155

Joint Underwater Mapping with Acoustic and Optical Images

Precious Philip-Ifabiyi

Zagreb, June 2025.

First and foremost, I offer my deepest gratitude to God Almighty for His infinite blessings and love. I humbly recognize that I could not have reached this milestone without His steady hand leading the way.¹

I would also like to express my heartfelt thanks to my family for their unwavering love, support, and encouragement throughout the highs and lows of this endeavor. You have always been my foundation, cheering me on and giving me the strength to believe in myself and my faith, even in moments of doubt.

To my supervisors, Dr. Nuno Gracias, Dr. Fausto Ferreira, and Valerio Franchi, I am immensely grateful for your invaluable guidance and insightful feedback throughout this thesis. Your mentorship has been instrumental to the successful completion of this work. I must extend a special note of gratitude to Dr. Gracias, who has not just been a supervisor but a true mentor; I feel privileged to have had him guide this work.

I also wish to extend my sincere appreciation to Prof. Pere Ridao for generously offering his time to brainstorm solutions to the challenges I encountered during my research.

To Hayat Rajani and Dr. Roger Pi, thank you for your kind assistance, which you offered so willingly despite your demanding schedules.

Finally, I want to acknowledge the incredible colleagues and lecturers I had the pleasure of knowing during this master's program. To my classmates, thank you for the shared laughter, the solidarity through struggle, and the unforgettable camaraderie. To my lecturers, thank you for teaching with such passion and for instilling in us the knowledge to make a meaningful impact.

¹Joshua 1:9

CONTENTS

1. Introduction	1
1.1. Background and Context	1
1.2. Problem Statement	2
1.3. Objectives and Scope	2
1.4. Thesis Contribution	3
1.5. Thesis Structure	4
2. Literature Review	5
2.1. Underwater Optical Imaging	5
2.1.1. Challenges in Underwater Optical Data Acquisition	5
2.1.2. Optical Feature Detection and Matching	7
2.2. Sonar Imaging	8
2.2.1. Feature Detection and Matching	8
2.2.2. Acoustic Mapping	9
2.3. Underwater Multimodal Imaging	10
2.3.1. Multimodal Feature Matching	10
2.3.2. Multimodal Mapping	11
2.4. Research Gaps	13
3. Methodology	14
3.1. Preliminaries on Sensor Modalities	14
3.1.1. Side-scan Sonar	14
3.1.2. Camera	17
3.2. System Overview and Data Acquisition	18
3.3. Feature Detection, Matching, and Tracking	19
3.3.1. Optical-Optical Feature Matching	19
3.3.2. Sonar-Sonar and Sonar-Optical Feature Matching	20
3.3.3. Feature Tracking and Selection	20

3.4. Multi-Session Optimization	21
3.4.1. Problem Formulation	21
3.4.2. Data Pre-processing	23
3.4.3. Factor Definitions and Error Models	23
3.4.4. Implementation Strategy: Keyframe Selection	28
3.4.5. Non-Linear Least-Squares Objective Function	28
3.4.6. Robust Cost Function for Optical Measurements	29
3.4.7. Solver	29
3.4.8. Initial Estimates	30
3.5. Post-processing and Map Generation	31
4. Results and Discussion	32
4.1. Experimental Data	32
4.2. Feature Matches	34
4.3. Optimization Results	37
4.3.1. Trajectories	37
4.3.2. Quantitative Analysis of Reprojection Errors	37
4.3.3. Qualitative Results	42
5. Conclusion	47
5.1. Limitations and Future Work	47
Bibliography	49

1. Introduction

1.1. Background and Context

The ocean covers over 70% of the Earth's surface, yet less than 30% of it has been comprehensively mapped. Accurate mapping and classification of seabed topography, structures, and habitats are crucial for numerous applications, including fundamental scientific research, maritime archaeology, and ecological monitoring.

Traditionally, bathymetric and seafloor imaging data are acquired using acoustic sensors. Multibeam echosounders (MBES) provide high-resolution bathymetry, while side-scan sonars (SSS) generate detailed acoustic backscatter imagery, revealing textural information about the seafloor. Sonar-based technologies are particularly essential for extensive surveys and operations in deep or turbid waters where optical methods are limited.

However, the interpretation of acoustic data, particularly SSS imagery, is a complex and labor-intensive process, typically performed manually and offline by domain experts in marine geology and biology. Acoustic imaging modalities possess inherent limitations compared to optical systems in clear conditions, such as lower spatial resolution, very different appearance of feature across different surveys, geometric distortions, and susceptibility to acoustic noise. All these factors degrade considerably the image quality. In addition, the acoustic signatures from different seafloor types are often not uniquely distinguishable, complicating classification tasks even for experienced analysts.

To augment the interpretability of acoustic data and improve the robustness of seafloor characterization, optical imaging serves as a valuable complementary modality. Typically deployed from Autonomous Underwater Vehicles (AUVs) or Remotely Operated Vehicles (ROVs), optical imaging systems capture high-resolution visual information. However, their effective range is constrained by light attenuation (absorption and scattering) and water turbidity. A significant operational challenge arises from vehicle localization inaccuracies. In the absence of continuous, precise external positioning (e.g., from Ultra-Short Baseline (USBL) systems, or unavailable GPS), the

dead-reckoning navigation systems inherent to AUVs accumulate drift over time. These inaccuracies lead to spatial misalignments between acoustic maps, which may possess their own georeferencing uncertainties, and optical imagery. Such misalignments, especially between datasets from different sessions, pose substantial hurdles for effective data fusion, comprehensive interpretation, and long-term monitoring.

1.2. Problem Statement

This thesis addresses the critical challenge of coherently aligning and integrating multi-modal (acoustic and optical) and multi-session underwater datasets for comprehensive seafloor mapping. The core problem lies in correcting spatial inconsistencies, arising from navigation drift, between side-scan sonar imagery and optical images, particularly when acquired during separate survey campaigns. The overarching goal is to develop a robust methodology capable of producing spatially accurate, consistent, and well-aligned composite maps. Such enhanced multimodal maps are essential for improving downstream tasks like automated seafloor classification by leveraging the synergistic strengths of both optical and acoustic sensing.

1.3. Objectives and Scope

This thesis is developed within the context of the IURBI project [1], which seeks to develop an intelligent AUV capable of real-time seafloor analysis and adaptive mission planning (Figure 1.1). A fundamental prerequisite for such autonomous capabilities is the ability to robustly align and fuse sensor data from multiple sources and surveys into a single, coherent model. This thesis addresses that foundational challenge by developing a comprehensive offline framework for multi-session, multimodal map alignment.

The primary objectives of this thesis are to:

- Develop a robust and flexible framework for the alignment and integration of side-scan sonar and optical imagery acquired in single or multiple sessions by AUVs, towfish, or ROVs.
- Formulate and implement a factor graph optimization approach to jointly refine vehicle trajectories and sensor alignments across multiple sessions and modalities, accommodating the inherent uncertainties in underwater navigation.
- Evaluate the performance of the proposed methodology using real-world underwater datasets, assessing its accuracy, robustness, and practical applicability.

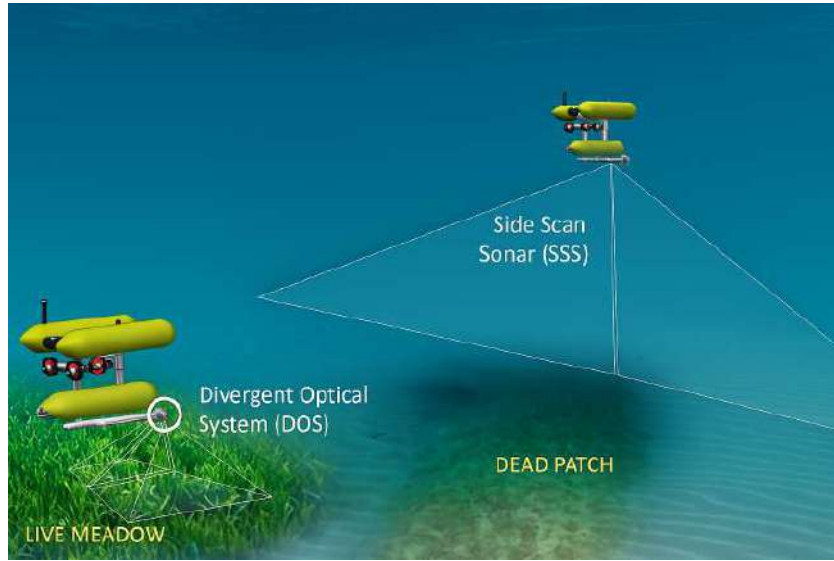


Figure 1.1: IURBI project concept. The AUV will monitor, among other types of seabed, live meadow, sand gaps, and dead leaves.

The scope of this work encompasses the offline processing and alignment of previously collected side-scan sonar and optical image datasets. While initial navigation data from the AUV/ROV is assumed to be available, this work specifically focuses on refining these initial pose estimates to achieve precise multimodal and multi-session co-registration.

1.4. Thesis Contribution

This research introduces a novel methodology for multimodal seafloor mapping that integrates side-scan sonar and optical imagery. To the best of our knowledge, this is the first approach that systematically addresses the spatial alignment and fusion of these two sensing modalities across multiple sessions. Furthermore, the proposed framework is adaptable for aligning datasets composed solely of side-scan sonar imagery from multiple sessions. Additionally, the framework is sensor-agnostic, which allows extension to other sensor types, provided that a sensor projection model and inter-measurement correspondences can be established.

The proposed framework enhances the interpretation and classification of the ocean floor and provides a foundation for further developments in autonomous underwater navigation and exploration.

1.5. Thesis Structure

The remainder of this thesis is organized as follows:

- Chapter 2 reviews the relevant literature on acoustic sensing, optical imaging, multimodal localization and mapping techniques.
- Chapter 3 presents the proposed methodology for aligning sonar and optical images, including data preprocessing, feature extraction, and the multi-session, multimodal optimization strategy.
- Chapter 4 discusses the experimental setup and data used, presents the results, and provides an analysis of the performance and limitations of the proposed approach.
- Chapter 5 concludes the thesis by summarizing key findings and suggesting directions for future research.

2. Literature Review

2.1. Underwater Optical Imaging

2.1.1. Challenges in Underwater Optical Data Acquisition

Acquiring high-quality optical data in underwater environments is fundamentally challenging due to the complex propagation of light through water. The primary factor degrading image quality is turbidity, caused by suspended particulate matter like sediment, plankton, and other organic or inorganic substances. These particles induce significant light scattering and absorption, which are quantified by the medium's inherent optical properties (IOPs). The IOPs govern the rapid wavelength-dependent attenuation of light, particularly in the red and infrared spectra, leading to severe color distortion and diminished intensity as depth or distance increases [2, 3]. Collectively, these phenomena reduce image contrast, obscure fine structural details, and degrade the performance of computer vision algorithms for tasks such as environmental mapping, object detection, and semantic interpretation. Furthermore, for deep water imaging where artificial lighting must be used, there are additional issues related to the fact that the light sources and the imaging sensors are in close proximity. Such proximity creates strong inhomogeneities in the illumination field which may result in under- or over-exposed portions of the images. It also gives rise to dynamically moving shadows, which are coupled to the motion of the acquisition setup.

To address these issues, extensive research has been dedicated to developing methods for underwater image restoration and enhancement. As categorized in comprehensive reviews of the field [4, 5, 6, 7], these techniques can be broadly divided into three paradigms: non-physical model-based methods, physical model-based methods, and learning-based approaches.

Non-physical model-based methods, often termed image enhancement techniques, operate directly on image properties like contrast and color without an explicit physical model of light transport. For instance, Iqbal et al. [8] introduced the Unsupervised

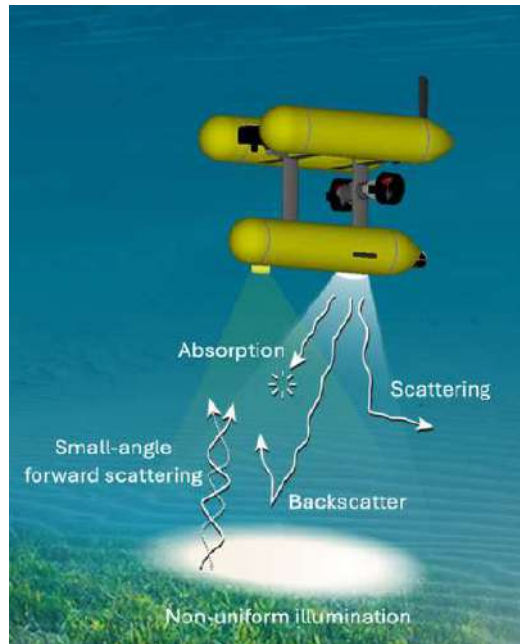


Figure 2.1: Image degradation in underwater environments

Colour Correction Method (UCM), which adjusts the RGB and HSI color spaces to correct color casts and improve contrast. This concept was later extended with modifications to histogram stretching [9]. Other notable approaches include the fusion of contrast-enhanced and color-corrected image versions [10] and the combined application of the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm on different color models [2]. While effective for improving visual appeal and the performance of subsequent feature detection, these methods are fundamentally limited in their ability to recover a physically accurate representation of the scene.

Physical model-based methods seek to mathematically invert the image degradation process to achieve true image restoration. These techniques are grounded in an understanding of underwater optics, most commonly approximated by the Jaffe-McGlamery model [11, 12]. This model describes the captured image as a combination of a directly transmitted signal, a forward-scattered signal (blur), and a backscattered signal (veiling light). The primary limitation of this approach is its reliance on accurate environmental parameters, which are often difficult to obtain in situ and may vary spatially across the scene. Notable recent work is that of Akkaynak and Treibitz [13] on a revision of the Jaffe-McGlamery model and the Sea-Thru method for removing water from underwater images [14].

Learning-based methods, particularly those employing deep learning, have emerged as a powerful data-driven alternative. These approaches can be trained on either

synthetic or real-world underwater imagery. Synthetic data-driven models, such as WaterGAN [15], leverage in-air RGB-D images and physical parameters to generate paired datasets of clear and degraded images for supervised training. While effective, these models often face a "sim-to-real" gap, struggling to generalize to the vast complexity of real underwater conditions. In contrast, real data-driven methods train on authentic underwater images. To overcome the scarcity of paired clear/degraded real images, many techniques employ unsupervised or weakly supervised frameworks. For example, cycle-consistent Generative Adversarial Networks (GANs) have been successfully used to learn the mapping between clear and degraded domains without requiring pixel-perfect pairs [16, 17, 18]. Despite their impressive results, the robustness and generalization of deep learning models in previously unseen underwater environments remain active areas of research.

2.1.2. Optical Feature Detection and Matching

Feature detection and matching are foundational components of many computer vision pipelines, including visual localization, 3D reconstruction, and Visual Simultaneous Localization and Mapping (VSLAM). The efficacy of these processes is severely tested in the optically challenging underwater domain. Historically, the field relied on hand-crafted feature detectors and descriptors. The Scale-Invariant Feature Transform (SIFT) established a benchmark with its invariance to scale and rotation [19], while Speeded-Up Robust Features (SURF) offered a computationally faster alternative [20]. However, the performance of these gradient-based methods deteriorates significantly in typical underwater imagery, which exhibits low contrast, blurriness, and non-uniform color distortion. These degradations alter the local image gradients upon which the descriptors are built, leading to a high rate of matching failures. To improve computational efficiency, binary descriptors were introduced. Methods such as BRIEF (Binary Robust Independent Elementary Features) [21] and its successor, ORB (Oriented FAST and Rotated BRIEF) [22], construct descriptors from simple intensity comparisons, enabling rapid matching via the Hamming distance. Despite its speed and widespread adoption in frameworks like ORB-SLAM, ORB's performance is often unstable in underwater scenes where low contrast and blur inhibit reliable keypoint detection, resulting in tracking failures [23]. Consequently, for optimal performance, handcrafted methods typically necessitate a preliminary image enhancement or restoration step.

To overcome the limitations of handcrafted features, recent research has shifted towards deep learning-based approaches. A primary obstacle is the scarcity of large-

scale, annotated underwater datasets required for training [23, 24, 25]. A prevalent and effective strategy to address this is the generation of synthetic training data. This involves either applying physical underwater imaging models to clear in-air image-depth pairs or using style transfer networks to simulate diverse underwater appearances [23, 24, 25]. Benefiting from these synthetic datasets, novel frameworks have been developed to produce more robust feature descriptors. A prominent paradigm is cross-modal knowledge distillation, utilized by systems like the Underwater Feature Extraction Network (UFEN) [23] and RU-SLAM's UWNet [25]. In this approach, knowledge from a powerful "teacher" model (e.g., SuperPoint, pre-trained on millions of in-air images) is transferred to a "student" network that learns to extract resilient features from the synthetic underwater images. Other strategies aim to directly learn attenuation-invariant features [24]. To manage the complex geometric and photometric distortions, these networks often integrate advanced architectural components such as channel attention modules, spatial attention modules, and deformable convolutions, which allow the model to adapt to spatially-variant degradation [24, 25]. While these learning-based methods demonstrate superior performance in recent literature, it is critical to note that their comparisons against conventional techniques like SIFT often omit the pre-enhancement step that is typically required for the fair evaluation of handcrafted methods [25].

2.2. Sonar Imaging

2.2.1. Feature Detection and Matching

Feature detection and matching in sonar imagery, particularly from side-scan sonar (SSS), is a cornerstone of applications like Simultaneous Localization and Mapping (SLAM) and seafloor mapping. This task, however, remains a formidable challenge due to low signal-to-noise ratios, significant geometric distortions, and intensity variations inherent in acoustic imaging.

Early research efforts logically focused on adapting well-established feature extraction algorithms from the computer vision domain. Techniques such as the Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), and Harris corners were applied to detect and describe keypoints in sonar images [26, 27, 28]. For instance, Kim et al. [28] utilized Harris corner points matched via cross-correlation to register acoustic images. In a comparative study, Vandrish et al. [26] evaluated several methods on SSS data and found that SIFT and a phase-correlation technique yielded the

most promising results. However, these conventional methods, which were designed for optical images, often struggle with the unique noise characteristics and textural properties of sonar data.

To address these shortcomings, subsequent research has focused on developing more specialized approaches. A notable example is the work by [29], who proposed a feature-matching algorithm based on a nonlinear scale space constructed using a nonlinear diffusion equation. This approach was specifically designed to preserve critical edge and texture information while suppressing noise, demonstrating superior matching accuracy compared to SIFT, SURF, KAZE, and ORB on their test data.

More recent approaches integrate sophisticated pre-processing and geometric constraints to handle the unique properties of SSS data. Zhang et al. [30], for instance, introduced a framework that first applies a canonical transformation to the raw images to correct for slant range distortions and normalize intensities based on a Lambertian reflectance model. For matching, they combine FAST corner detection with SIFT descriptors. Critically, the correspondence search is constrained by a near-neighbor search guided by geo-referenced data and a sliding compatibility check to enforce geometric consistency along the survey track. A significant limitation of this method, however, is its reliance on a flat seafloor assumption for the canonical transform and the availability of accurate odometry, which may not hold in complex terrains or with significant dead-reckoning drift.

Building on the principle of leveraging sensor geometry, Yang et al. [31] proposed a framework for non-consecutive data association (NDA) to achieve robust matching across large viewpoint changes. In their method, keypoints are first identified as pixels with high gradients. The core of the NDA module is an epipolar line search, where initial pose errors from odometry are refined using the Iterative Closest Point (ICP) algorithm. This geometry-aided approach was shown to outperform both conventional descriptors (SIFT, SURF, ORB) and a learning-based method (LightGlue [32]). However, their final matching step relies on minimizing intensity error along the calculated epipolar line. This assumption can be violated due to complex acoustic scattering effects that cause the intensity of a physical landmark to vary significantly with the observation angle.

2.2.2. Acoustic Mapping

The use of SSS images in Simultaneous Localization and Mapping (SLAM) has been an active research area for over two decades [33, 34, 30, 35]. Aulinas et al. [33]

and Zhang et al. [30] employed feature-based SLAM techniques to extract and track features across multiple SSS images. While [33] used an Extended Kalman Filter (EKF)-based approach, [30] adopted pose-graph optimization, demonstrating improved trajectory estimation. Zhang et al. [35] further introduced a subframe-based dense SLAM framework that integrates SSS data to optimize AUV trajectory and seafloor mapping. Their results indicated that incorporating constraints from SSS images effectively reduced localization drift.

Beyond SSS-based SLAM, researchers have also investigated feature-based SLAM frameworks leveraging forward-looking sonars (FLS) [36, 37].

2.3. Underwater Multimodal Imaging

2.3.1. Multimodal Feature Matching

The fusion of sonar and optical imagery for underwater mapping has garnered significant attention, primarily due to the complementary nature of these sensing modalities. Existing methodologies for multimodal feature matching can be broadly categorized into three main approaches: (1) transformation-based methods that convert images into a common domain prior to feature matching; (2) direct feature extraction techniques that identify correspondences across modalities without domain transformation; and (3) projection-based methods that project features from one modality into the coordinate system of another.

The first category focuses on transforming one imaging modality to replicate the style of another, thereby simplifying the feature matching process. Abu et al. [38] proposed a framework that transforms optical images into Synthetic Aperture Sonar (SAS) images, followed by handcrafted feature descriptor extraction for classification. Jang et al. [39] developed a style-transfer algorithm using a VGG-19 network to convert acoustic images into an optical style before applying the SIFT descriptor for matching. While innovative, this method's validation was limited to a single test case, raising questions about its generalizability. Building on this, Zhou et al. [40] extended the approach by modifying the VGG-19 network to perform bidirectional transformations between acoustic and optical images. Their method utilized HardNet descriptors for feature matching, which reportedly yielded superior results compared to SIFT. However, their evaluation was also conducted on a limited dataset, highlighting the need for broader testing to confirm its robustness. In a subsequent work, Jang et al. [41] enhanced their original approach by integrating SuperGlue [42] for more robust

feature matching between the style-transferred sonar image and the optical image, which significantly increased the quantity and quality of correspondences. This body of work underscores the potential of style transfer but also illustrates the need for evaluation across more diverse underwater environments to establish broader applicability.

The second category of research bypasses domain transformation in favor of direct cross-modal feature matching. Liu et al. [43] introduced a scale-adaptive matching algorithm for optical and forward-looking sonar (FLS) images, which combined iterative image enhancement with a correlation filter (MOSSE) and Gaussian scale-space. This technique effectively addressed noise, scale, and angular variations, achieving high matching success rates. However, its evaluation in a controlled pool environment revealed performance degradation with poor image quality or in the presence of acoustic shadows, suggesting limitations for real-world marine applications. While learning-based feature matchers have demonstrated significant promise in other cross-modal tasks (e.g., Optical-SAR, RGB-Event) [44, 45], their application to optical-acoustic matching remains largely unexplored.

The third category leverages the known intrinsic and extrinsic sensor parameters to project features between modalities. This approach is highly dependent on accurate sensor calibration and projection models. Cardaillac and Ludvigsen [46] developed a system that uses a multibeam FLS to resolve the scale ambiguity inherent in monocular camera systems. Their method accounts for the sonar's lack of elevation data by representing a sonar feature as a vertical line segment on the camera's image plane. A visual feature from the SLAM-generated point cloud is considered a match if its 2D projection falls on this sonar-derived segment. Qiu et al. [47] employ a similar concept but fuse sonar data with features from a stereo camera to obtain precise depth information. Instead of projecting a single point, their method defines a 3D spatial region corresponding to a strong sonar echo and associates all visual features within this volume as a group of "candidate" matches. In a different approach, Zhang et al. [48] integrate a stereo camera, imaging sonar, and an IMU within a tightly-coupled SLAM framework. Their method projects detected visual landmarks into the 2D sonar image coordinate system and establishes a match by finding the closest sonar landmark whose Euclidean distance falls below a predefined threshold.

2.3.2. Multimodal Mapping

This section reviews key works in multimodal mapping and SLAM that specifically integrate acoustic and optical sensors for underwater applications.

The inherent limitations of single-sensor systems in dynamic underwater environments have driven the development of multi-sensor fusion techniques. While vision-based systems excel in clear water, they falter in turbid or low-light conditions where acoustic sensors remain robust. On the other hand, images produced by acoustic sensors have lower resolution compared to those of vision-based systems [49].

Hover et al. [50] demonstrated an integrated system for autonomous ship hull inspection that combines data from an imaging sonar and a monocular camera. Their approach is notable in that it does not perform direct feature matching between the two sensor modalities. Instead, the sensors operate concurrently, often imaging different parts of the hull, and their measurements are fused as separate constraints within a unified pose-graph SLAM framework (iSAM). This back-end fusion allows for real-time, drift-free vehicle control relative to the previously unseen hull structure by leveraging the strengths of each sensor independently.

Kunz [51] developed a framework for AUV navigation and mapping in unstructured environments, such as coral reefs, by fusing data from multibeam sonar and stereo cameras. The method solves for vehicle trajectories by incorporating constraints from both sensor types into a factor graph. Visual constraints are derived from landmark-based reprojection error, and sonar constraints produce relative pose error. A key contribution is the inclusion of sensor offsets as variables to be estimated within the graph, improving the precision of the final map.

Zhang et al. [48] propose a multisensor fusion underwater SLAM method that tightly couples a stereo camera, a multibeam imaging sonar, and an IMU. The system comprises two collaborative subsystems, a visual-inertial and an acoustic-inertial system, which enhances robustness by allowing one to function independently if the other fails. A unified factor graph jointly optimizes measurements from all sensors, minimizing distinct residual errors for IMU, visual, and sonar data to achieve continuous and stable localization even in instances of visual failure.

Cardaillac and Ludvigsen [46] developed a SLAM framework that utilizes a multi-beam FLS to correct the scale ambiguity of a monocular V-SLAM system by estimating a depth ratio. This ratio is stabilized over time using Maximum Likelihood Estimation (MLE) on a set of correspondences. Although the camera-sonar combination significantly improves localization, the final 3D map reconstruction in their work relies solely on the rescaled point cloud from the monocular camera.

Extending their earlier work on feature matching, Jang et al. [41] integrated their opti-acoustic matching into a multi-session SLAM framework. While Negahdaripour et al. [52] conducted preliminary work on opti-acoustic stereo imaging, Jang et al. further

developed and implemented an opti-acoustic pairwise factor, utilizing iSAM [53] as the backend for SLAM optimization. Inspired by the work of Kim et al. [54], anchor nodes were used in the factor graph to estimate the relative transformation between the acoustic session and the camera session. However, this work was demonstrated using a DIDSON FLS, and its applicability to side-scan sonar (SSS) based mapping has not been explored.

Other notable tightly coupled systems like SVIn2 fuse mechanical scanning profiling sonar, visual, inertial, and depth data to achieve robust performance (including initialization, loop closing, and relocalization) under challenging underwater conditions such as haze, low light, and motion blur [55].

2.4. Research Gaps

Despite the advancements in multimodal underwater mapping, a comprehensive review of the literature reveals several unaddressed areas. No existing work was found that focuses specifically on the fusion of side-scan sonar (SSS) imagery with optical images for joint mapping and SLAM. The majority of current research targets forward-looking sonar (FLS) systems.

Furthermore, while significant progress has been made in feature matching between sonar and optical data, a universally robust method that performs reliably across diverse underwater conditions and sensor configurations has yet to emerge. The development of such a method remains a critical challenge for the field.

3. Methodology

This chapter details the proposed methodology for multi-session, multimodal underwater mapping using optical and side-scan sonar (SSS) imagery. The core of this work is a framework designed to produce a coherent, globally-aligned 2D map by optimizing vehicle poses and landmark positions from multiple survey sessions. The system pipeline, illustrated in Figure 3.4, is composed of three main stages: (1) feature detection and matching across and within sensor modalities, (2) temporal feature tracking, and (3) a final optimization stage based on factor graphs to refine all state variables simultaneously. Each component of this pipeline is explained in the subsequent sections. Prior to this, we provide an overview of the sensor modalities used in the system.

3.1. Preliminaries on Sensor Modalities

3.1.1. Side-scan Sonar

A side-scan sonar (SSS) is an acoustic sensing device designed to capture echo intensity profiles of the seafloor. It is commonly mounted on a towfish, which can be deployed from a ship or integrated into AUVs or ROVs. Figure 3.1 illustrates the geometric model of a typical SSS system and includes relevant symbols used throughout this section.

The system consists of two transducer arrays positioned on the port and starboard sides of the vehicle. These arrays are oriented perpendicularly to the direction of travel and emit fan-shaped ultrasonic pulses at regular intervals. Each pulse, also referred to as a ping, has a narrow beam in the horizontal plane and a wide beam in the vertical plane (Figure 3.2). When emitted, the pulse ensonifies a region of the seafloor, referred to as the ensonified region (ER). The reflected acoustic energy, or backscatter, is then recorded by the sonar head at discrete time intervals, forming a vector of intensity values known as a swath. Each element of this swath, which corresponds to a specific time delay and thereby a specific range, is called a bin.

Given the speed of sound in water, denoted as c , and the time of flight for a returned

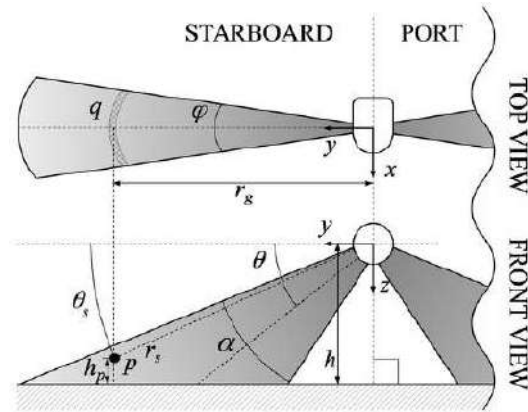


Figure 3.1: Side-scan sonar model. The x-axis represents the vehicle's motion direction, also known as the along-track direction. [56]

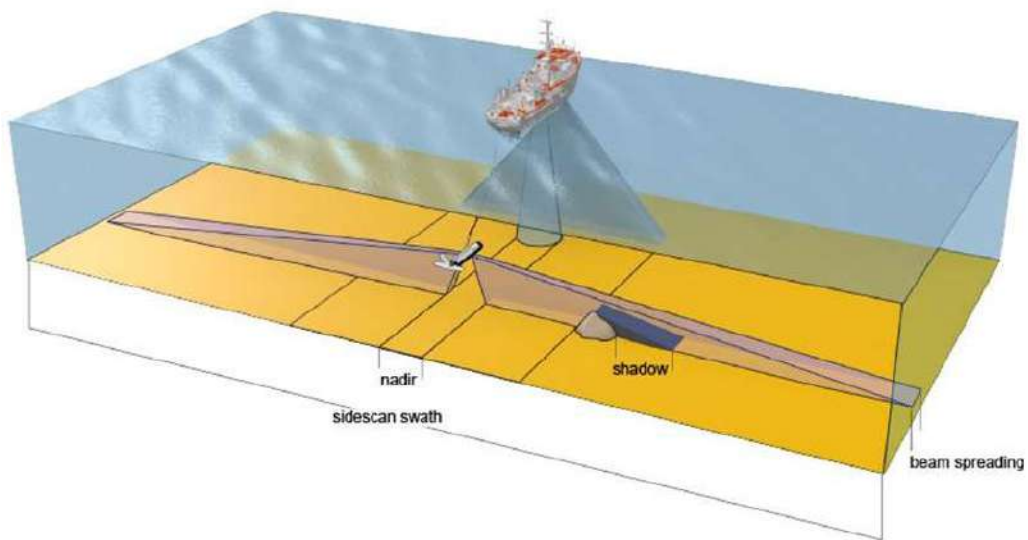


Figure 3.2: Typical deployment of a side-scan sonar, installed in a towfish and towed from a survey vessel.

echo, t , the slant range r_s , which is the distance from the sonar head to the reflecting point on the seafloor, can be calculated using:

$$r_s = c \cdot \frac{t}{2} \quad (3.1)$$

While the speed of sound in water is known to vary with temperature, salinity, and pressure, a constant value for c is assumed in this work. However, it is crucial to acknowledge the implications of this assumption. If the actual speed of sound deviates significantly from the assumed constant value, the calculated slant range measurements will be inaccurate. Specifically, an overestimation of the speed of sound will lead to an overestimation of the slant range, while an underestimation will result in an underestimated slant range. This can introduce errors in the mapping of the seafloor and affect the accuracy of subsequent calculations and interpretations.

The resolution along the slant range is determined by the sampling frequency of the sonar, i.e., the number of bins per ping. This resolution must be appropriately configured based on the vehicle's speed. Higher speeds necessitate higher sampling rates to maintain image fidelity and minimize geometric distortion.

To locate a point, p , within the YZ plane corresponding to a bin in the swath, polar coordinates (r_s, θ_s) are used. The grazing angle θ_s , which defines the angle between the incident beam and the local tangent to the seafloor, can be computed from the vehicle altitude h , the point's altitude h_p , and the slant range r_s as follows:

$$\theta_s = \arcsin\left(\frac{h - h_p}{r_s}\right) \quad (3.2)$$

The vehicle altitude h can be obtained using external sensors such as a Doppler Velocity Log (DVL) or estimated from sonar data using techniques such as the First Bottom Return (FBR). However, estimating the point altitude h_p solely from SSS data is highly challenging, and often infeasible without auxiliary bathymetric information. As a result, the grazing angle θ_s is typically constrained within a range defined by the sensor's vertical beam width α and mounting angle θ :

$$\theta_s \in \left[\theta - \frac{\alpha}{2}, \theta + \frac{\alpha}{2}\right] \quad (3.3)$$

To infer the 3D position of a point p , many approaches adopt the flat seabed assumption, which presumes that the seafloor within the ensonified region is flat (i.e., $h_p = 0$). This assumption simplifies the inversion of sonar measurements but limits reconstruction accuracy. In this work, we aim to relax this assumption and instead

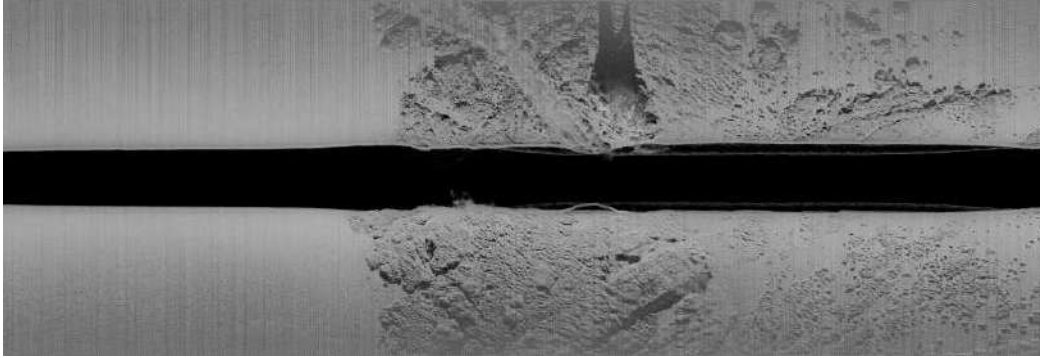


Figure 3.3: Example of an acoustic image (waterfall) generated by side-scan sonar.

estimate the 3D location of points using a factor graph optimization framework, as detailed in Section 3.4.

In the XY plane, a similar uncertainty arises regarding the exact location of the point p along the arc of constant slant range q . Since the echo in a bin may result from one or more scatterers within this arc, precise localization is ambiguous. However, this uncertainty is generally neglected based on the assumption that the horizontal beam is sufficiently narrow. This is reasonable given that the beam's horizontal opening angle ϕ typically ranges from 0.2° to 1.5° .

The two sonar heads operate simultaneously and at the same frequency, producing swaths that can be combined side-by-side to form a complete cross-track profile. As the vehicle moves forward, successive swaths are stacked vertically to construct an acoustic image. In this image, each pixel corresponds to a bin in the swath. Figure 3.3 shows an example of such an image, often referred to as a waterfall image. Here, darker tones represent weaker echo intensities, while brighter tones indicate stronger reflections. A distinct data gap, or nadir, typically appears as a dark band in the center of the image, separating the port and starboard channels. This zone exists because the transducers are angled downward at an angle (θ), preventing the area of the seafloor directly beneath the towfish from being ensonified. The time it takes for the first bottom return to be received can also contribute to an increase in the width of this dark band.

For a comprehensive understanding of side-scan sonar principles, we direct the reader to [57, 58, 59, 60].

3.1.2. Camera

Given the prevalent understanding of optical camera geometry and image formation, a detailed exposition is omitted here. Comprehensive treatments of these topics are

available in [61, 5] for reference.

3.2. System Overview and Data Acquisition

The proposed system is designed to process datasets collected during separate survey sessions, potentially involving different underwater platforms. An overview of the data flow is presented in Figure 3.4. The pipeline is initiated with sensor data and initial dead-reckoning pose estimates from each session. This includes waterfall representations of SSS data and optical camera frames. The system uses three types of feature matches: sonar-sonar, optical-optical, and the critical sonar-optical cross-modal matching. These matches are organized into feature tracks that can exist within and across sessions. In the scope of this thesis, a feature track is a collection of observations, acoustical or optical, of the same 3D point in the seafloor. Finally, a multi-session, multimodal optimization backend processes these tracks to yield optimized vehicle poses and landmark positions, culminating in a unified 2D multimodal map.

The datasets utilized in this work were gathered in a shallow-water marine environment characterized by seagrass beds of *Posidonia oceanica* [62] intertwined areas of sand and hard-bottom. A more in-depth description of the datasets is provided in Chapter 4. Data collection was performed using two platforms in multiple sessions:

- A towfish, equipped with a side-scan sonar and an integrated attitude and pressure sensor (providing roll, pitch, heading, and depth), was towed by a surface vessel. The towfish’s navigation solution was achieved by combining multiple data sources: a vessel-mounted Real-Time Kinematic (RTK) GPS provided a high-accuracy global position for the ship, while an Ultra-Short Baseline (USBL) acoustic positioning system measured the towfish’s submerged position relative to the vessel. The attitude sensor provided direct orientation measurements for the submerged platform.
- An Autonomous Underwater Vehicle (AUV), equipped with two down-looking optical cameras, was used to capture high-resolution imagery of the seafloor. The AUV’s navigation solution was provided by an onboard Integrated Navigation System (INS) aided by a Doppler Velocity Log (DVL) and a pressure sensor for depth.

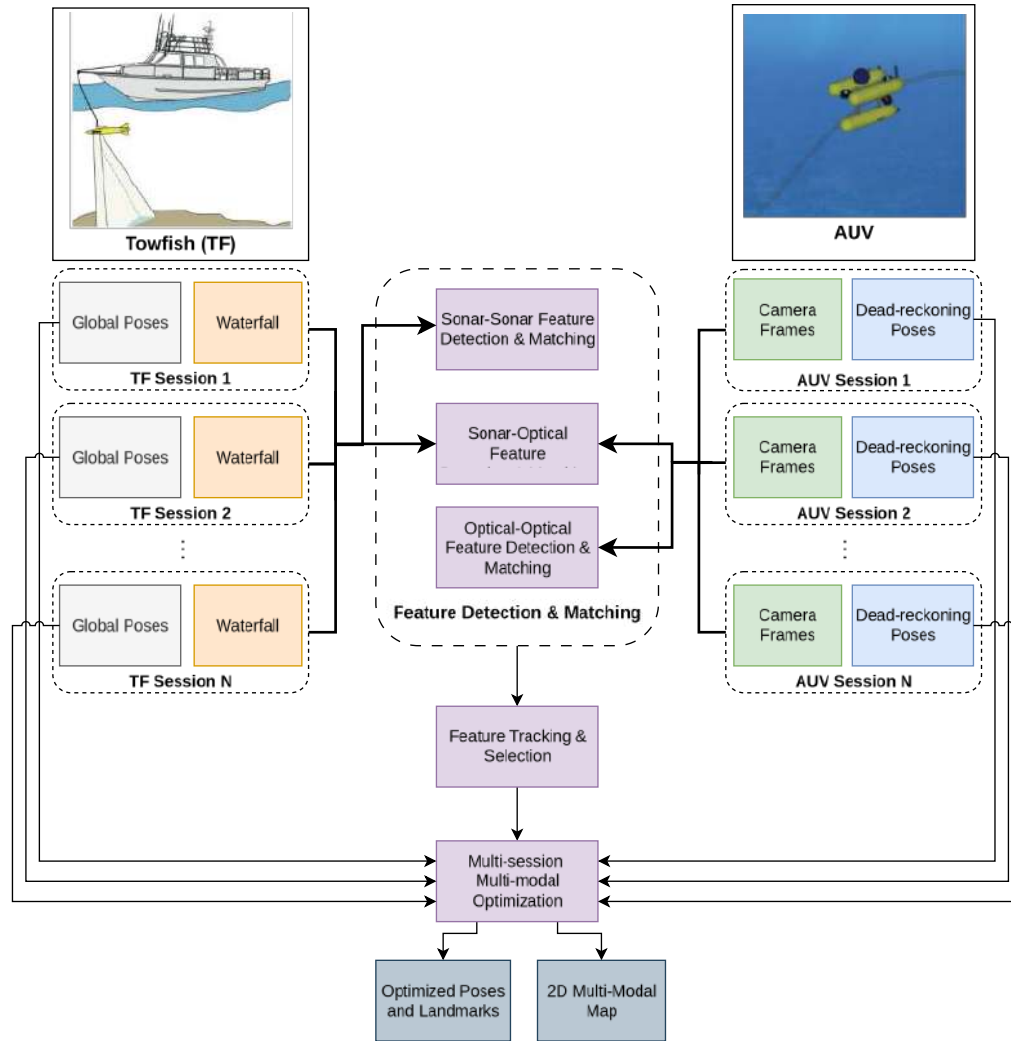


Figure 3.4: An overview of the proposed system pipeline, illustrating the data flow from filtered sensor inputs through feature matching and final optimization to produce a multimodal map.

3.3. Feature Detection, Matching, and Tracking

A fundamental component of this framework is the establishment of robust data associations (features) both within a single data stream (unimodal) and between different sensor types (multimodal).

3.3.1. Optical-Optical Feature Matching

For registering consecutive optical images, features are detected and described using the Scale-Invariant Feature Transform (SIFT) algorithm. Given that the data was collected near the seafloor with adequate artificial illumination, no explicit image pre-processing

was performed. SIFT features are matched between temporally sequential images to establish local motion constraints. Then, RANSAC is used to remove outliers in the matches.

3.3.2. Sonar-Sonar and Sonar-Optical Feature Matching

Due to the significant difference in data phenomenology and the presence of high-texture, low-feature regions like seagrass, robust automated feature matching between sonar and optical images, as well as between two sonar images, remains an open research challenge. Consequently, correspondences for both sonar-sonar and sonar-optical pairs were established manually for this work. Matching was performed between salient, recognizable landmarks and topographical edges present in both the SSS waterfall display and the corresponding optical frames.

3.3.3. Feature Tracking and Selection

Following pairwise matching, features are organized into tracks that represent the same 3D landmark observed over time. A feature track can consist of observations from a single modality (e.g., optical-only or sonar-only) or from multiple modalities (a mix of optical and sonar observations). These tracks form the basis of the constraints used in the global optimization. To enhance the robustness of this process, a selection mechanism is applied to the optical-only feature tracks.

Optical-Only Feature Track Filtering Algorithm

To improve the quality of the data associations and reduce computational overhead, a selection process is used to retain only the most reliable optical-only feature tracks. The core principle is to prioritize tracks that are both long, indicating feature stability, and well-distributed throughout the image sequence.

The selection criterion is as follows: for each image in a session, we identify the N feature tracks with the greatest total length (i.e., the highest number of observations) that appear in that image. For this work, N was set to 4. The final set of optical-only tracks used for the optimization is the union of these top N tracks from every image.

This strategy effectively discards short, potentially spurious tracks while ensuring that every part of the trajectory is constrained by its most persistent and reliable visual features.

3.4. Multi-Session Optimization

3.4.1. Problem Formulation

To jointly optimize the vehicle trajectories, landmark positions, and sensor extrinsics, we formulate the problem as a factor graph, as depicted in Figure 3.5. This graphical model allows for the flexible fusion of heterogeneous information sources. The graph, shown in Figure 3.5, consists of variable nodes representing the quantities to be estimated and factor nodes representing probabilistic constraints on these variables, derived from sensor measurements and motion models.

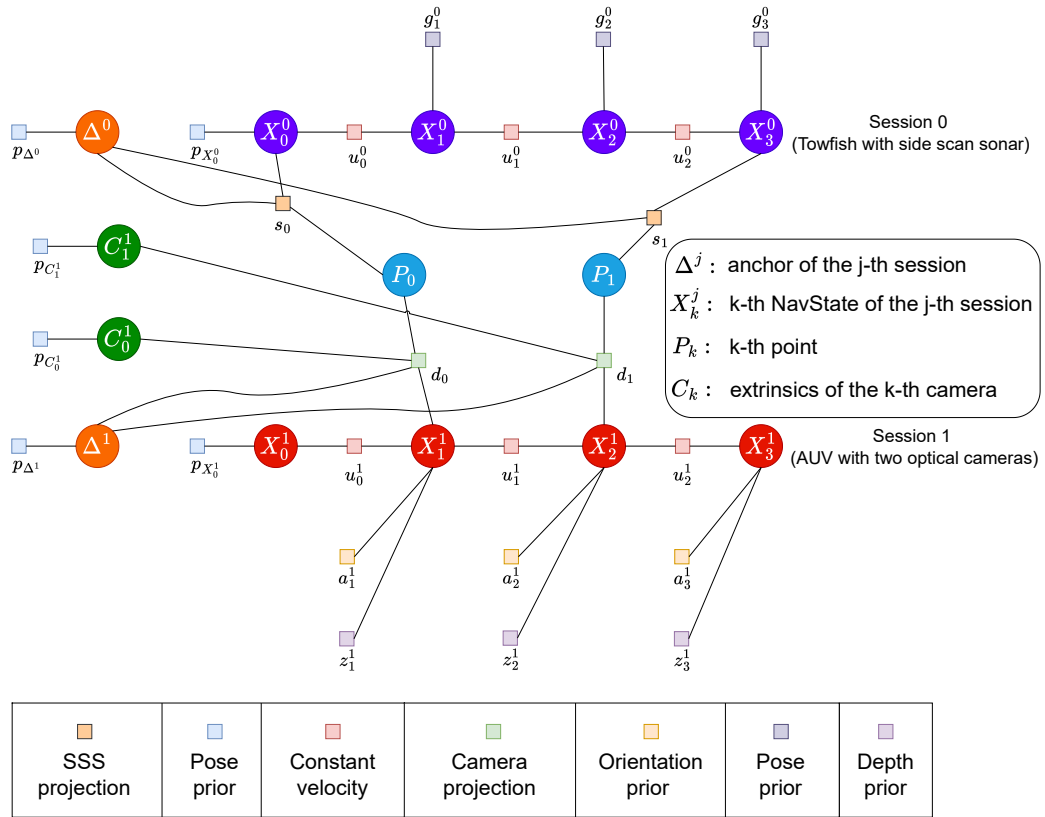


Figure 3.5: The factor graph representation of the proposed multi-session, multimodal mapping framework. It connects vehicle states (X_k^j), landmarks (P_k), and session anchors (Δ^j) through various measurement and motion factors.

The primary variables to be estimated are:

- $\Delta^j \in SE(3)$: The anchor node representing the global pose of session j .
- $X_k^j \in SE(3) \times \mathbb{R}^3$: The 9-DOF navigation state of the vehicle at time k in session j , composed of the vehicle's pose $T_k^j \in SE(3)$ (which contains position p_k^j and

orientation R_k^j) and its linear velocity $v_k^j \in \mathbb{R}^3$.

- $P_k \in \mathbb{R}^3$: The 3D position of landmark k in the world frame.
- $C_m^j \in SE(3)$: The extrinsic calibration pose of camera m in AUV session j .

We adopt the concept of an anchor node for each session to align data from different sessions into a common global reference frame [54, 63]. The anchor, Δ^j , represents the global transformation of the j -th session's local coordinate frame relative to the world frame.

The goal is to find the set of state variables that maximizes the posterior probability given all measurements and priors Z , which corresponds to the Maximum a Posteriori (MAP) estimate. Let the complete set of variables to be optimized be $\Theta = \{\{\Delta^j\}, \{X_k^j\}, \{P_k\}, \{C_m^j\}\}$, where C_m^j is the extrinsic calibration for the m -th camera in the j -th AUV session. The MAP estimate, Θ^* , is the set of variables that maximizes the posterior probability given all measurements and priors, Z :

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} P(\Theta|Z) \quad (3.4)$$

Based on the factor graph formulation, this posterior is proportional to the product of all prior probabilities and measurement likelihoods. Let J_T be the set of N_T towfish sessions and J_A be the set of N_A AUV sessions, with $J = J_T \cup J_A$. The posterior probability is given by:

$$\begin{aligned} P(\Theta|Z) \propto & \underbrace{\prod_{j \in J} P(\Delta^j | p_{\Delta^j}) \cdot \prod_{j \in J} P(X_0^j | p_{X_0^j})}_{\text{Session Anchor \& Initial State Priors}} \\ & \cdot \underbrace{\prod_{j \in J_T} \prod_k P(X_k^j | g_k^j)}_{\text{Towfish Global Pose Priors (e.g., GPS/USBL)}} \\ & \cdot \underbrace{\prod_{j \in J_A} \left(\left(\prod_m P(C_m^j | p_{C_m^j}) \right) \cdot \prod_k P(X_k^j | a_k^j) \cdot \prod_k P(X_k^j | z_k^j) \right)}_{\text{AUV Priors (Extrinsics, Orientation, Depth)}} \\ & \cdot \underbrace{\prod_{j \in J} \prod_k P(X_k^j | X_{k-1}^j, u_{k-1}^j)}_{\text{Constant Velocity Motion Model}} \\ & \cdot \underbrace{\prod_{j \in J_T} \prod_k P(s_k^j | X_{i_k}^j, P_{l_k}, \Delta^j)}_{\text{SSS Landmark Measurements}} \\ & \cdot \underbrace{\prod_{j \in J_A} \prod_k P(d_k^j | X_{i_k}^j, P_{l_k}, C_m^j, \Delta^j)}_{\text{Camera Landmark Measurements}} \end{aligned} \quad (3.5)$$

3.4.2. Data Pre-processing

Side-Scan Sonar Slant Range Estimation

A key pre-processing step is to convert the raw side-scan sonar keypoints, identified in the waterfall image space by a triplet '(ping_number, side, bin_index)', into metric slant range measurements. The 'side' parameter specifies whether the feature was observed on the port (left) or starboard (right) channel of the sonar for a given ping. The 'bin_index' does not directly correspond to a range. Instead, the slant range, r_k , for a keypoint is determined via linear interpolation based on the maximum recorded slant range, $R_{\max}$, for that specific ping and side. The calculation is:

$$r_k = R_{\max} \cdot \frac{\text{bin\_index}_k}{\text{bin\_index}_{\max}} \quad (3.6)$$

The resulting SSS measurement vector, z_{SSS} , used in the optimization is therefore $z_{\text{SSS}} = [r_k, 0]^T$. The second element represents the zero along-track distance, as the measurement is assumed to lie on a line perpendicular to the vehicle's direction of travel.

3.4.3. Factor Definitions and Error Models

This section details the mathematical models for the factors used in the optimization.

Prior Factors

Prior factors are used to incorporate initial estimates or absolute measurements into the factor graph. They constrain single variables, which helps anchor the solution, prevent drift, and ensure convergence.

Initial State and Anchor Priors These factors initialize the optimization by providing an absolute reference for each session. A prior is placed on the first state of each session, X_0^j , and on the session's anchor pose in the world frame, Δ^j .

The error for the initial state prior, $e_{X_0}(\cdot)$, is a 9-dimensional vector composed of a 6-DOF pose error and a 3-DOF velocity error. The error is computed by treating each component appropriately: the pose error is calculated on the $SE(3)$ manifold, while the velocity error is calculated in the R^3 space. Given a prior mean $p_{X_0^j}$ (composed of pose $\bar{T}_{X_0^j}$ and velocity $\bar{v}_{X_0^j}$), the error function is:

$$e_{X_0}(X_0^j, p_{X_0^j}) = \begin{bmatrix} \log((\bar{T}_{X_0^j})^{-1} T(X_0^j))^{\vee} \\ v(X_0^j) - \bar{v}_{X_0^j} \end{bmatrix} \quad (3.7)$$

where $T(\cdot)$ and $v(\cdot)$ extract the pose and velocity from the state, respectively. For the $SE(3)$ anchor pose, the error is calculated similarly using the logarithmic map relative to its prior mean $\bar{p}_{\Delta^j}$:

$$e_{\Delta}(\Delta^j, \bar{p}_{\Delta^j}) = \log((\bar{p}_{\Delta^j})^{-1} \Delta^j)^{\vee} \quad (3.8)$$

Camera Extrinsic Prior To provide a good initial guess and constrain the optimization, a prior is placed on the extrinsic calibration, C_m^j , for each camera m in each AUV session j . The error function $e_C(\cdot)$ is defined similarly on the $SE(3)$ manifold:

$$e_C(C_m^j, \bar{p}_{C_m^j}) = \log((\bar{p}_{C_m^j})^{-1} C_m^j)^{\vee} \quad (3.9)$$

where $\bar{p}_{C_m^j}$ is the initial estimate for the camera's pose.

Absolute Measurement Priors These unary factors are critical for correcting drift during a trajectory.

- **Global Pose Factors:** For towfish sessions, a full $SE(3)$ pose measurement g_k^j (from fused GPS/USBL and attitude data) provides an absolute constraint on the state X_k^j . The error $e_g(\cdot)$ is:

$$e_g(X_k^j, g_k^j) = \log((g_k^j)^{-1} T(X_k^j))^{\vee} \quad (3.10)$$

where $T(X_k^j)$ extracts the full $SE(3)$ pose from the state.

- **Orientation and Depth Factors:** A key aspect of our methodology is to model the constituent state variables according to their underlying physical observability, as reported by the AUV's onboard navigation filter. The filtered pose data is not monolithic in its error characteristics; a distinction must be made between components subject to unbounded drift and those that are globally referenced.

The AUV's horizontal position (x, y) is estimated via dead reckoning, integrating velocity measurements (e.g., from a DVL and IMU). This process is inherently subject to integration drift, causing the uncertainty in horizontal position to grow without bound over time in the absence of external corrections like GPS or acoustic positioning.

In contrast, other state components have bounded error characteristics. The vehicle's depth (z) is directly measured by a pressure sensor, providing an absolute measurement relative to the water surface based on hydrostatic pressure. Similarly, the vehicle's 3D orientation is globally referenced. The roll and pitch angles are determined with high accuracy by the Inertial Measurement Unit

(IMU), which measures the direction of the constant gravity vector. The yaw, or heading, is observed by a magnetometer referencing the Earth's magnetic field. Given this dichotomy, our factor graph formulation treats these components differently. The drifting horizontal states are constrained by relative motion factors (as described in Section 3.4.3), while the bounded-error states are constrained by unary factors (priors) at each time step. These unary factors anchor the state estimate to the absolute measurements provided by the filter. The corresponding residual functions for depth and orientation are:

$$e_a(X_k^j, a_k^j) = \log(R(X_k^j)^{-1} \cdot a_k^j)^\vee \quad (3.11)$$

$$e_z(X_k^j, z_k^j) = z(X_k^j) - z_k^j \quad (3.12)$$

where $R(\cdot)$ and $z(\cdot)$ extract the orientation and depth from the state variable.

Constant Velocity Factor

To ensure trajectory smoothness and to propagate state estimates between keyframes, we incorporate a constant velocity motion model. This factor places a constraint between two consecutive vehicle states, X_{k-1}^j and X_k^j . The error function, $e_u(\cdot)$, represents the deviation from the constant velocity assumption, expressed in the local body frame of the second state, X_k^j . The error is first computed in the world frame and then rotated into the body frame using the orientation of the second pose, R_k^j .

$$e_u(X_{k-1}^j, X_k^j) = \begin{bmatrix} (R_k^j)^T \left((p_{k-1}^j + v_{k-1}^j \Delta t) - p_k^j \right) \\ (R_k^j)^T (v_{k-1}^j - v_k^j) \end{bmatrix} \quad (3.13)$$

where the top block is the position error and the bottom block is the velocity error, both rotated into the body frame of state X_k^j .

This error is modeled as a zero-mean Gaussian, $e_u(\cdot) \sim \mathcal{N}(0, \Sigma_u)$. The 6x6 covariance matrix Σ_u is derived from a continuous white-noise acceleration uncertainty vector, $\sigma_a = [\sigma_{ax}, \sigma_{ay}, \sigma_{az}]^T$, which represents independent uncertainties along each vehicle axis. The resulting covariance is block-diagonal:

$$\Sigma_u = \begin{bmatrix} \Sigma_p & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & \Sigma_v \end{bmatrix} \quad (3.14)$$

where Σ_p and Σ_v are diagonal matrices containing the variances for the position and velocity components, respectively:

$$\Sigma_p = \text{diag} \left(\left(\frac{1}{2} \sigma_{ax} \Delta t^2 \right)^2, \left(\frac{1}{2} \sigma_{ay} \Delta t^2 \right)^2, \left(\frac{1}{2} \sigma_{az} \Delta t^2 \right)^2 \right) \quad (3.15)$$

$$\Sigma_v = \text{diag}((\sigma_{ax}\Delta t)^2, (\sigma_{ay}\Delta t)^2, (\sigma_{az}\Delta t)^2) \quad (3.16)$$

where Δt is the time interval between the states.

Camera Projection Model

The camera factor relates a 3D landmark P_{l_k} to its 2D pixel observation d_k^j . Its error function, $e_{\text{cam}}(\cdot)$, is defined as the difference between the observed and predicted pixel coordinates:

$$e_{\text{cam}}(\Delta^j, X_{i_k}^j, C_m^j, P_{l_k}, d_k^j) = d_k^j - h_{\text{cam}}(\Delta^j, X_{i_k}^j, C_m^j, P_{l_k}) \quad (3.17)$$

The projection function, $h_{\text{cam}}(\cdot)$, first transforms the landmark from world coordinates into the local 3D Cartesian frame of the camera, ${}^C_m P_{l_k}$:

$${}^C_m P_{l_k} = \mathcal{H}^{-1} \left((C_m^j)^{-1} \cdot (T_{i_k}^j)^{-1} \cdot (\Delta^j)^{-1} \cdot {}^W \bar{P}_{l_k} \right) \quad (3.18)$$

where:

- ${}^C_m P_{l_k}$: is the 3D Cartesian point $[X, Y, Z]^T$ in the local frame of camera m .
- $\mathcal{H}^{-1}(\cdot)$: is the dehomogenization operator.
- $C_m^j \in SE(3)$: is the extrinsic pose of camera m relative to the vehicle base in session j .
- $T_{i_k}^j \in SE(3)$: is the vehicle's local pose, extracted from the 9-DOF state $X_{i_k}^j$.
- $\Delta^j \in SE(3)$: is the anchor pose for session j .
- ${}^W \bar{P}_{l_k}$: is the homogeneous representation of the world landmark P_{l_k} .

This 3D point is then projected to pixel coordinates using the standard pinhole camera projection model:

$$h_{\text{cam}}(\cdot) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \quad (3.19)$$

where f_x, f_y are the camera's focal lengths and c_x, c_y are the principal point coordinates. The error is modeled as a zero-mean Gaussian, $e_{\text{cam}}(\cdot) \sim \mathcal{N}(0, \Sigma_{\text{cam}})$, where Σ_{cam} is the 2x2 measurement noise covariance, typically assumed to be diagonal:

$$\Sigma_{\text{cam}} = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

The terms σ_u and σ_v represent the standard deviation of the measurement uncertainty in the horizontal and vertical pixel directions, respectively.

Side-Scan Sonar Projection Factor

This factor relates a 3D landmark P_{l_k} to its 2D measurement $s_k^j = [r_k, 0]^T$. The error function is:

$$e_{SSS}(\Delta^j, X_{i_k}^j, P_{l_k}, s_k^j) = s_k^j - h_{SSS}(\Delta^j, X_{i_k}^j, P_{l_k}) \quad (3.20)$$

The projection function, $h_{SSS}(\cdot)$, first transforms the landmark into the sonar's local reference frame:

$${}^S P_{l_k} = \mathcal{H}^{-1} \left(({}^B T_S)^{-1} \cdot (T_{i_k}^j)^{-1} \cdot (\Delta^j)^{-1} \cdot {}^W \bar{P}_{l_k} \right) \quad (3.21)$$

where:

- ${}^S P_{l_k}$: is the 3D Cartesian point in the sonar's local frame.
- ${}^B T_S \in SE(3)$: is the extrinsic pose of the sonar sensor relative to the vehicle base.
- Δ^j , $T_{i_k}^j$, and ${}^W \bar{P}_{l_k}$ are defined as in the camera projection model.

From this 3D point, the function predicts the slant range and along-track distance:

$$h_{SSS}(\cdot) = \begin{bmatrix} \|{}^S P_{l_k}\| \\ ({}^S P_{l_k})_x \end{bmatrix} \quad (3.22)$$

where $\|\cdot\|$ is the Euclidean norm, and $(\cdot)_x$ extracts the x-component of the vector. The error is modeled as a zero-mean Gaussian, $e_{SSS}(\cdot) \sim \mathcal{N}(0, \Sigma_{SSS})$, where Σ_{SSS} is the 2x2 measurement noise covariance, typically assumed to be diagonal:

$$\Sigma_{SSS} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_{at}^2 \end{bmatrix} \quad (3.23)$$

The variance components are not fixed but are calculated adaptively for each measurement. The stochastic model is explicitly designed to translate the inherent feature location uncertainty from the 2D sonar waterfall image into a metric covariance. This image-space uncertainty is resolved into two orthogonal components:

- Range Variance (σ_r^2): The variance for the range component quantifies the feature location uncertainty in the cross-track direction of the waterfall image. It is calculated by converting this pixel uncertainty into a metric variance using the sonar's known range resolution:

$$\sigma_r^2 = (\eta_r \cdot \rho_r)^2 \quad (3.24)$$

where η_r is the noise parameter representing the standard deviation in range pixels, and ρ_r is the sonar's range resolution (e.g., meters per pixel).

- Along-Track Distance Variance (σ_{at}^2): The variance for the zero along-track distance constraint is modeled as the sum of two independent error sources: the direct along-track uncertainty from feature localization and a geometric uncertainty from the sonar's physical beam width.

$$\sigma_{at}^2 = (\eta_a \cdot d_a)^2 + (z_r \cdot \theta_{bw})^2 \quad (3.25)$$

The first term quantifies the direct along-track uncertainty, where η_a is the standard deviation in pixels for the along-track direction and d_a is the local along-track resolution (i.e., the average forward displacement of the vehicle per pixel row in the waterfall). The second term models the geometric uncertainty in the along-track direction caused by the sonar's horizontal beam width, θ_{bw} . This uncertainty grows with the measured slant range, z_r , as the beam spreads.

3.4.4. Implementation Strategy: Keyframe Selection

To ensure computational tractability, a keyframe-based approach is employed rather than adding every vehicle state to the factor graph. A dense trajectory, particularly from a high-frequency sonar, can contain tens of thousands of poses, making the corresponding optimization problem prohibitively large. Therefore, the graph is constructed using only a sparse set of "key states" that are most informative.

A state is selected as a key state if it meets one of two criteria: (1) it has one or more landmark observations (either optical or sonar) associated with it, or (2) it is added at a fixed time interval. The first criterion ensures that all measurement constraints are anchored to a state in the graph, while the second criterion ensures a sufficient number of poses are included to accurately represent the trajectory's shape and enforce smoothness via the motion model.

3.4.5. Non-Linear Least-Squares Objective Function

Under the assumption that all noise sources are Gaussian, the MAP estimation problem is equivalent to solving a non-linear least-squares problem. This is achieved by minimizing the negative log-likelihood of the posterior probability. For a Gaussian distribution, this is equivalent to minimizing a sum of squared Mahalanobis distances.

The squared Mahalanobis distance for an error vector e with a given covariance matrix Σ is defined as:

$$\|e\|_{\Sigma}^2 \triangleq e^T \Sigma^{-1} e \quad (3.26)$$

Using the error functions defined in Section 3.4.3, the final objective function to be minimized is therefore expressed as the sum of the squared Mahalanobis distances of all factor errors:

$$\begin{aligned} \Theta^* = \underset{\Theta}{\operatorname{argmin}} \bigg(& \sum_{j \in J} \left(\|e_{\Delta}(\Delta^j, p_{\Delta^j})\|_{\Sigma_{\Delta}}^2 + \|e_{X_0}(X_0^j, p_{X_0^j})\|_{\Sigma_{X_0}}^2 \right) \\ & + \sum_{j \in J_A} \sum_m \|e_C(C_m^j, p_{C_m^j})\|_{\Sigma_C}^2 + \sum_{j \in J_T} \sum_k \|e_g(X_k^j, g_k^j)\|_{\Sigma_g}^2 \\ & + \sum_{j \in J_A} \sum_k \left(\|e_a(X_k^j, a_k^j)\|_{\Sigma_a}^2 + \|e_z(X_k^j, z_k^j)\|_{\Sigma_z}^2 \right) \\ & + \sum_{j \in J} \sum_k \|e_u(X_{k-1}^j, X_k^j)\|_{\Sigma_u}^2 \\ & + \sum_{j \in J_T} \sum_k \|e_{SSS}(\Delta^j, X_{i_k}^j, P_{l_k}, s_k^j)\|_{\Sigma_{SSS}}^2 \\ & + \sum_{j \in J_A} \sum_k \|e_{\text{cam}}(\Delta^j, X_{i_k}^j, C_m^j, P_{l_k}, d_k^j)\|_{\Sigma_{\text{cam}}}^2 \bigg) \end{aligned} \quad (3.27)$$

3.4.6. Robust Cost Function for Optical Measurements

The feature matching between optical images is performed automatically and is therefore susceptible to outliers from incorrect data associations. To ensure these outliers do not corrupt the optimization result, a robust cost function is applied to the camera projection factors. The standard squared-error term for camera measurements in Equation 3.27 is replaced by the Huber loss, $\rho(\cdot)$:

$$\sum_{j \in J_A} \sum_k \rho \left(\|e_{\text{cam}}(\Delta^j, X_{i_k}^j, C_m^j, P_{l_k}, d_k^j)\|_{\Sigma_{\text{cam}}}^2 \right) \quad (3.28)$$

The Huber loss behaves quadratically for small errors but linearly for large errors, effectively down-weighting the influence of outliers. It is defined as:

$$\rho(s) = \begin{cases} \frac{1}{2}s, & \text{if } s \leq \delta^2 \\ \delta \left(\sqrt{s} - \frac{1}{2}\delta \right), & \text{if } s > \delta^2 \end{cases} \quad (3.29)$$

where s is the residual error and δ is a tuning parameter that defines the threshold between inliers and outliers.

3.4.7. Solver

The non-linear least-squares problem defined by Equation 3.27 is solved using the Georgia Tech Smoothing and Mapping (GTSAM) library [64], which implements an efficient factor graph-based optimization using the Levenberg-Marquardt algorithm.

3.4.8. Initial Estimates

Non-linear least-squares solvers like the Levenberg-Marquardt algorithm require initial estimates for all variables in the factor graph. A good initial guess helps the algorithm converge faster and more reliably to the global or a meaningful local minimum. The initialization strategies for each variable type are detailed below.

- **Vehicle State (X_k^j):** The trajectory for each session is initialized in its own local reference frame. The first state, X_0^j , is set to the identity pose ($T = \mathbf{I}$) with zero velocity ($v = \mathbf{0}$). Subsequent states, X_k^j , are then computed relative to this origin using the raw navigation data. The initial velocity for each state is estimated via finite differencing of the position components between consecutive poses.
- **Anchors (Δ^j):** The initial estimate for each session anchor, Δ^j , is set to the first georeferenced pose provided by the raw navigation data for that session. This value also serves as the mean for a weakly-informative (loose) prior factor on the anchor node, allowing the optimizer flexibility while still grounding the solution globally.
- **Camera Extrinsics (C_m^j):** Although the precise extrinsic calibration is unknown, an initial estimate for each camera pose, C_m^j , is available. This estimate is used to initialize the variable and to define a relatively tight prior. This constrains the optimization and prevents convergence to physically implausible values.
- **Landmarks (P_k):** The initialization strategy for a 3D landmark depends on the type of observations in its feature track:
 - **Sonar-Only Tracks:** For tracks containing only SSS observations, each measurement is projected onto a plane representing the seafloor, using the initial vehicle pose and altitude. The initial landmark position, P_k , is then set to the centroid (average) of these projected points.
 - **Camera-Only Tracks:** Standard triangulation is used to compute the initial 3D position from multiple camera observations.
 - **multimodal Tracks:** The position is estimated as a weighted average of the sonar-only estimate and the camera-only estimate. The weights are proportional to the number of observations from each modality within the track.
 - **Special Cases:** If a multimodal track contains only a single camera view, triangulation is not possible, and only the sonar-based estimate is used. If a camera-only triangulation results in a cheirality exception (i.e., the

point is triangulated behind a camera), the landmark is initialized at the origin (0,0,0) to be corrected by the optimizer.

3.5. Post-processing and Map Generation

Once the optimization has converged, a series of post-processing steps are performed to generate the final dense trajectories and map products.

Dense Trajectory Interpolation The optimization yields a sparse set of poses corresponding to key states. To reconstruct the full, dense trajectory, poses for intermediate states are interpolated. This process uses the underlying geometry by treating poses as elements of the Special Euclidean group, $SE(3)$.

Interpolation is performed along the geodesic path on the manifold, which represents the shortest constant-velocity motion between two poses. For any two consecutive poses, T_1 and T_2 , the relative transformation between them is first mapped to the Lie algebra $\mathfrak{se}(3)$ via the logarithmic map. This yields a single vector representing the entire rigid motion. This vector is then scaled according to the desired interpolation timestamp and mapped back from the Lie algebra to the $SE(3)$ manifold via the exponential map. The resulting transformation is applied to the first pose to obtain the final interpolated pose, ensuring a smooth and physically plausible trajectory.

Map Generation The final, dense, globally-aligned trajectories are used to generate georeferenced maps for each sensor modality.

- **Optical Mosaic:** The optical images and their corresponding optimized global poses are processed using MosaicViewer [65] to generate a high-resolution, georeferenced optical map of the surveyed area.
- **Sonar Mosaic:** The side-scan sonar data is processed using MB-System [66] to produce a georeferenced GeoTIFF image of the acoustic backscatter.

4. Results and Discussion

4.1. Experimental Data

The proposed methodology was tested on optical and side-scan sonar data of the seabed collected during surveys along the Catalan coast (Figure 4.2). The optical images were obtained by using the Girona 1000 AUV, equipped with two down-looking cameras (Figure 4.1a). The cameras were mounted in a divergent stereo configuration with an angle of approximately 60 degrees between the camera's axes. The field of view of the cameras allows for an overlap of approximately 20% between the two cameras. The reason for choosing this configuration was to have a relatively high angle of view for both port and starboard, while assuring that there were no visual gaps in the area below the vehicle. The sonar data was collected by Tecnoambiente S.L., an environmental consultancy company, using a Klein System 3000 Side Scan Sonar [67] deployed from a survey vessel (Figure 4.1b). The localization system of the vehicles is discussed in Section 3.2.

The acoustic data was collected in October 2021, while the optical data was collected in April 2024. The time difference between the two collection efforts is relatively large and was due to vehicle availability and other logistics. It is expectable that some changes may have occurred during this period. However, as can be seen later in this section, the seafloor structures that were matched between the two modalities are quite distinctive and have similar geometric appearance (taking into consideration the distinct imaging characteristics of the two sensors).



Figure 4.2: General view of the study area where the data used in this thesis was collected, in the North part of the Catalan coast, in Spain. This figure covers approximately 20 km of coast line and is centered in the town of Sant Feliu de Guíxols. The light blue outlines represent the side-scan sonar transects used in this thesis, collected as part of coastal survey conducted by Tecnoambiente S.L. covering water depths between 10 and 40 m (marked in red).

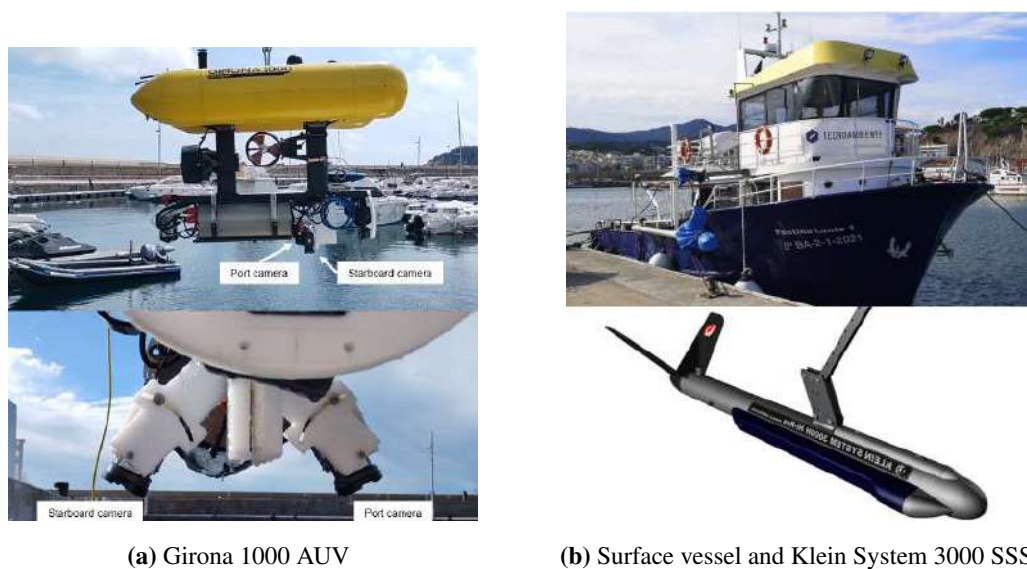


Figure 4.1: Vehicles used for data collection

Type of Matches	Number of matches
Sonar-Sonar	68
Sonar-Optical	22
Optical-Optical	14345

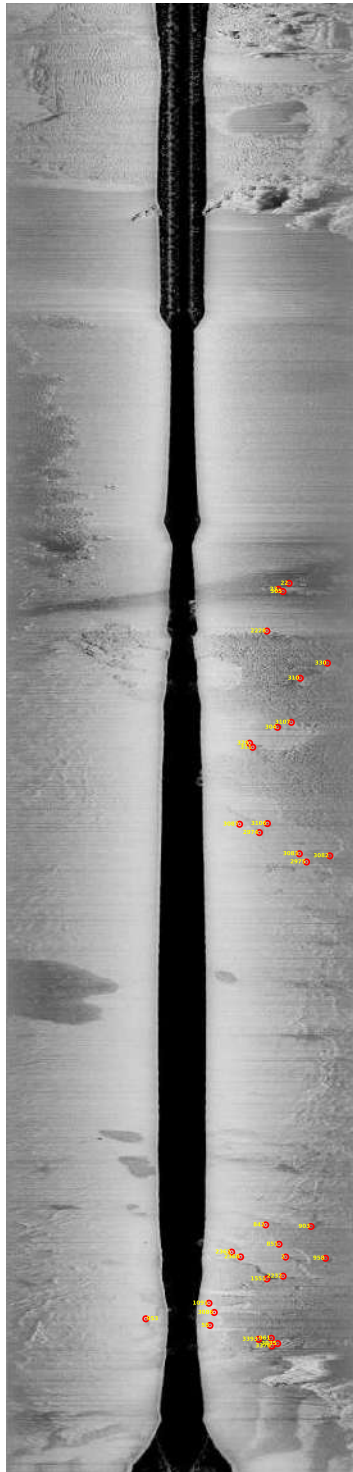
Table 4.1: Number of matches

Five sessions, comprising four sonar sessions (illustrated in Figures 4.3 and 4.4) and one optical session, from surveys conducted in the same geographical area, were utilized to evaluate the proposed framework. Henceforth, the four sonar sessions will be referred to as *SES0*, *SES1*, *SES2*, and *SES3*, respectively, and the optical session will be referred to as *SES4*. Given that there is no ground-truth localization, the proposed methodology will be evaluated quantitatively (using outputs from the optimization) and qualitatively.

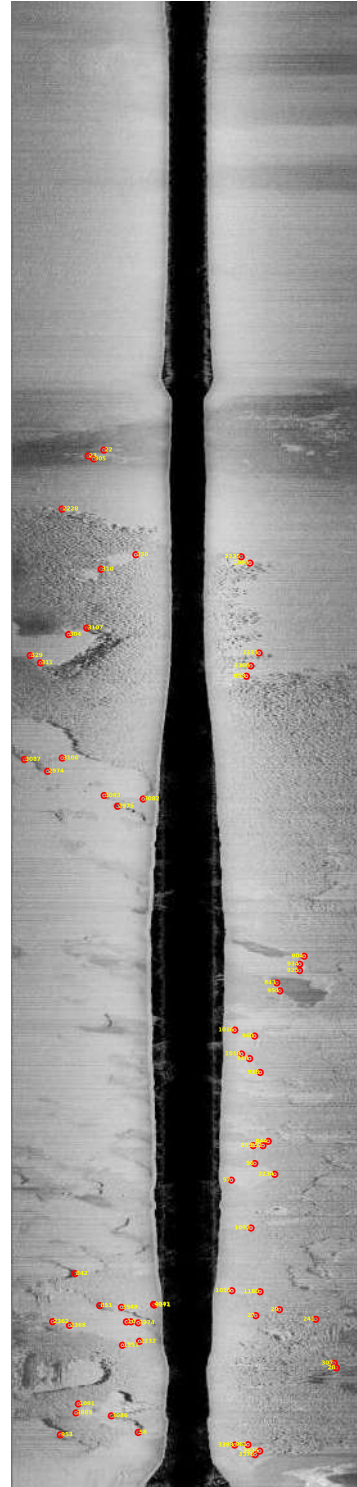
4.2. Feature Matches

The distribution of uni-modal and multimodal matches used for the optimization is shown in Table 4.1. As can be seen from the table, the number of optical-optical matches is large, compared to the others. This is because SIFT was used for automatic feature detection and matching, and it produced many matches, even though the seabed had repetitive features (seagrass leaves). It can also be observed that the number of matches between sonar and optical images is comparatively low, which can be attributed to the difficulty in observing unique and distinguishable matches between the two modalities by the human annotator. Matches were mainly made on contours of the seabed regions and visible landmarks (such as on distinctive transitions from seagrass to sand and on salient objects such as cement blocks used for boat anchoring).

Figure 4.3 shows the waterfall representation of the data collected during all the sonar sessions, with features shown in red. These features are numbered such that the same feature seen in another session will share the same number. It can be observed that the features are generally located around contours of geomorphic features. We can also observe the non-uniform distribution of the features along the images.

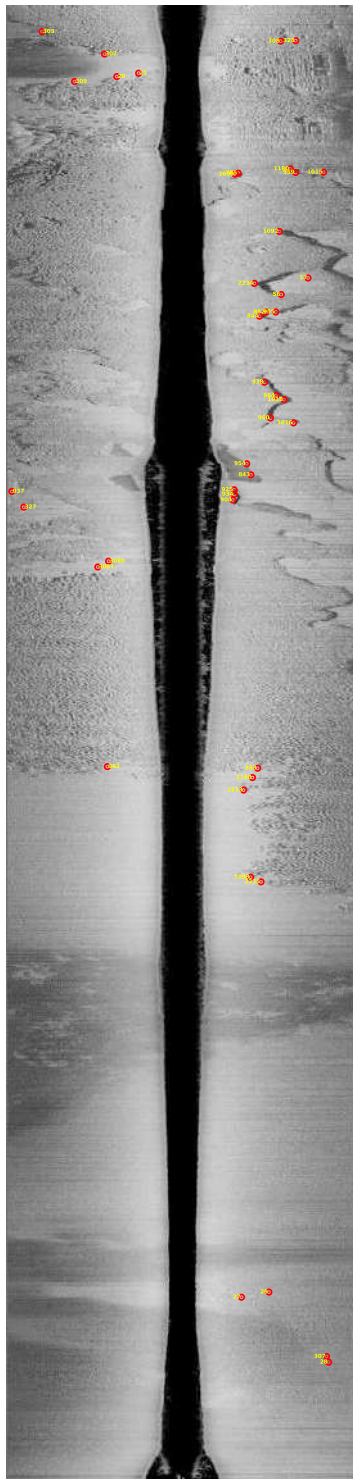


(a) SES0

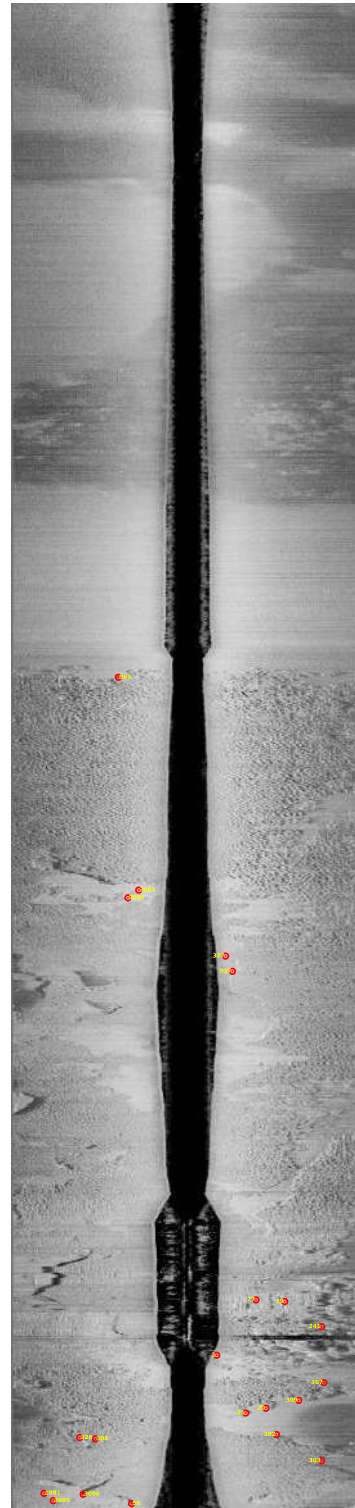


(b) SES1

Figure 4.3: Waterfall images of SES0 and SES1 with keypoints shown in red and their corresponding track IDs in yellow. To focus on the relevant areas, pings from SES0 that do not overlap with other sessions have been omitted from this view.



(a) SES2



(b) SES3

Figure 4.4: Waterfall images of the SES2 and SES3 with keypoints shown in red and keypoint track number in yellow.

4.3. Optimization Results

The final factor graph, comprising 15,258 variable nodes and 31,901 factors, was optimized to produce the final state estimates. The optimization process converged in approximately 5.9 seconds on a laptop equipped with a 2.3 GHz Intel Core i7 processor. This chapter presents a qualitative and quantitative analysis of the results, focusing on the refined vehicle trajectories, the reduction in sensor projection errors, and the final generated maps.

4.3.1. Trajectories

The trajectories taken by the vehicles before and after optimization are shown in Figure 4.5. We can observe that while the navigation data of the individual sonar sessions is accurate, a global transformation shift was performed by the optimizer to align all the sessions' trajectories. With the sessions' anchors, the optimizer was able to make this shift. Moreover, it can also be observed that there are local deformations in the trajectory of the sonar sessions, particularly in places with point constraints. The majority of these local deformations are smooth, which suggests the effectiveness of the constant velocity factor in ensuring that the dynamics of the towfish is taken into consideration.

The optimized points observed in the sonar sessions are also shown in Figure 4.5. We do not show the optimized points with only optical-optical matches due to their large size.

4.3.2. Quantitative Analysis of Reprojection Errors

This section presents a quantitative analysis of the framework's performance by evaluating the projection errors before and after optimization.

The boxplots in Figure 4.6 show the error magnitudes for four distinct categories of feature tracks. A significant reduction in both the median error and the overall error spread is evident across all categories, except the plot on sonar reprojection error for multimodal tracks.

The analysis of camera-only tracks in the boxplot (Figure 4.6) reveals a number of outliers, despite the low median error after optimization. Further investigation indicates that these large errors are generally not due to incorrect feature matching. Instead, they are attributed to a lack of convergence for specific 3D landmarks within the optimization, caused by a poor initial guess of their 3D position. A primary cause for these poor initial estimates is cheirality exceptions that occur during the initial triangulation of a point. A

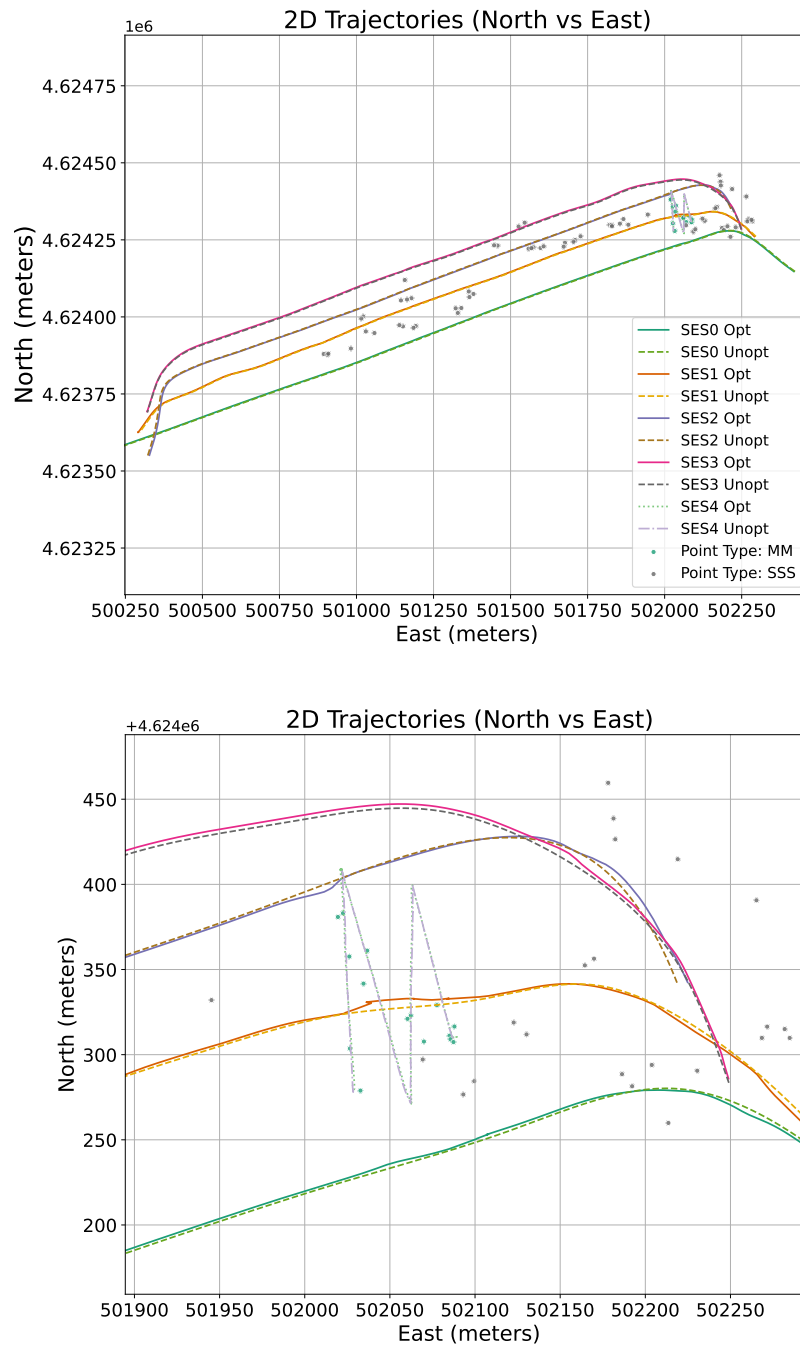


Figure 4.5: X-Y plots of the vehicles' trajectories before and after optimization, marked in dashed and full lines respectively. The long parallel trajectories are the four SSS sessions and the shorter zig-zagged trajectory is the optical session. The dots represent the optimized positions of the 3D points, in grey for the sonar-to-sonar matches and in green for the optical-to-sonar matches. The lower figure shows a zoomed-in section of the upper figure.

cheirality exception happens when the triangulated 3D point mathematically lies behind one of the observing cameras, which is physically impossible. In our implementation, when such an exception is detected, the landmark is assigned a default initial position of (0,0,0) in the world frame. This default position can be very far from the true landmark location, providing a difficult starting point from which the non-linear optimizer may fail to converge to the correct solution. Figure 4.9 shows examples of such feature tracks that are correctly matched across images but still result in a large final reprojection error due to this initialization and convergence issue.

Another key observation is that the reduction in median error is more pronounced for camera measurements than for sonar measurements within multimodal tracks. This is an expected outcome of the system's error modeling. Since the sonar data has a lower spatial resolution than the optical imagery, its associated measurement uncertainty (Σ_{SSS}) was set higher than that of the camera (Σ_{cam}). Consequently, during optimization, the solver places more weight on the high-precision camera measurements, driving their errors down more aggressively. Moreover, the majority of the multimodal feature tracks contained only one camera observation, which made the initial estimate of the 3D point to be estimated using the sonar observations and the flat sea-bed assumption.

The distribution of the projection errors provides further insight into the performance of the optimization. The histogram of camera reprojection errors, shown in Figure 4.7, compares the initial and final error distributions. After optimization, the final error distribution is visibly more compact, indicating a significant reduction in variance. Notably, the histogram of the final errors closely follows a zero-mean Gaussian distribution, which suggests that the camera projection model has minimal systematic bias.

The 2D error distributions, visualized in the scatter plots of Figure 4.8, offer additional details. For the sonar projection error (Figure 4.8a), the optimization yields a final error distribution that is tightly clustered around the origin. A key characteristic of this final distribution is its anisotropy, showing a greater variance in the along-track dimension than in the range dimension. This is an expected outcome, as the adaptive noise model correctly accounts for the larger uncertainty in the along-track direction arising from the sonar's physical beam width.

The corresponding scatter plot for the camera reprojection error (Figure 4.8b) confirms the presence of the previously discussed outliers, which are attributed to poor initial 3D point estimates. Despite these outliers, the plot clearly shows that the vast majority of errors are contained within a very small radius around the origin, reinforcing the low-variance and low-bias nature of the final solution for the camera measurements.

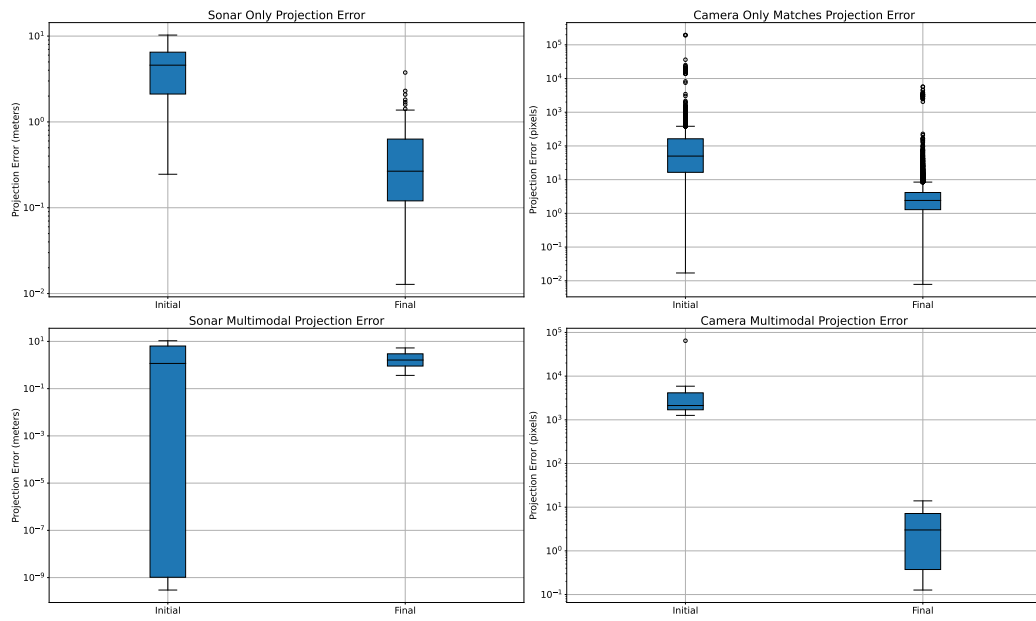


Figure 4.6: Boxplots comparing the initial and final scalar projection errors, defined as the Euclidean distance between observed and projected measurements. The vertical axis in these plots is represented in logarithmic scale. The analysis is presented for both sonar (error in meters) and camera (error in pixels) data, and is further categorized by uni-modal (e.g., Sonar Only) and multimodal feature tracks. The whiskers on each plot extend to 1.5 times the inter-quartile range (IQR).

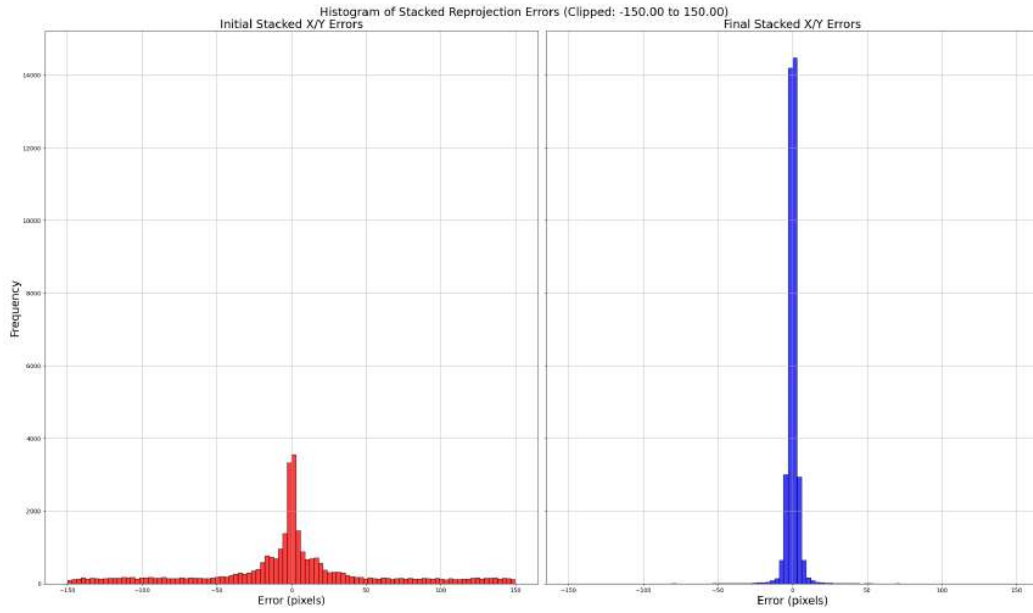


Figure 4.7: Initial and final histograms of the all camera reprojection residues, including both the u and v image dimensions. The axes of the plots are kept constant for comparison purposes.

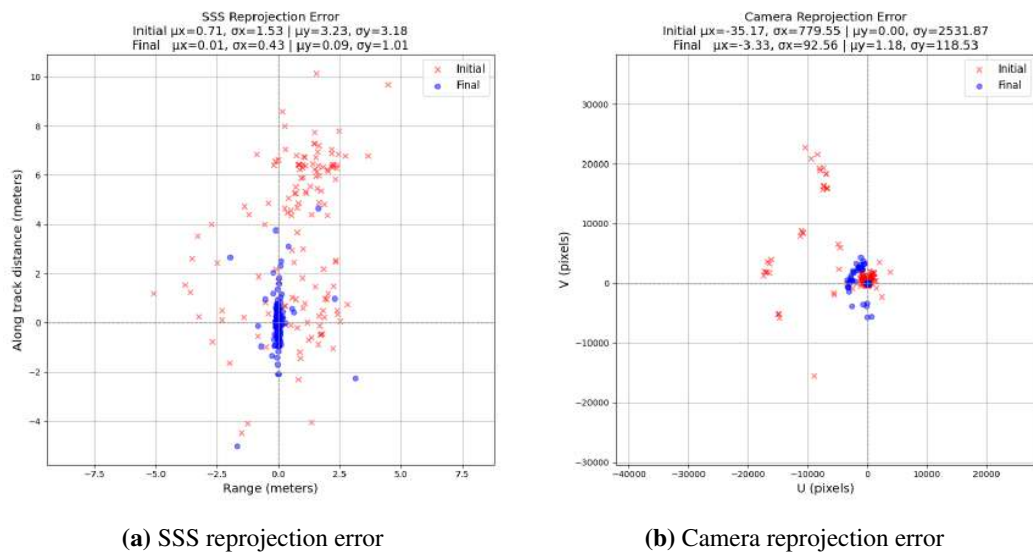


Figure 4.8: Scatter plots of the initial and final projection errors for sonar and camera observations. For visual clarity, the plot axes have been focused on the main error distribution, and extreme outliers are not shown.

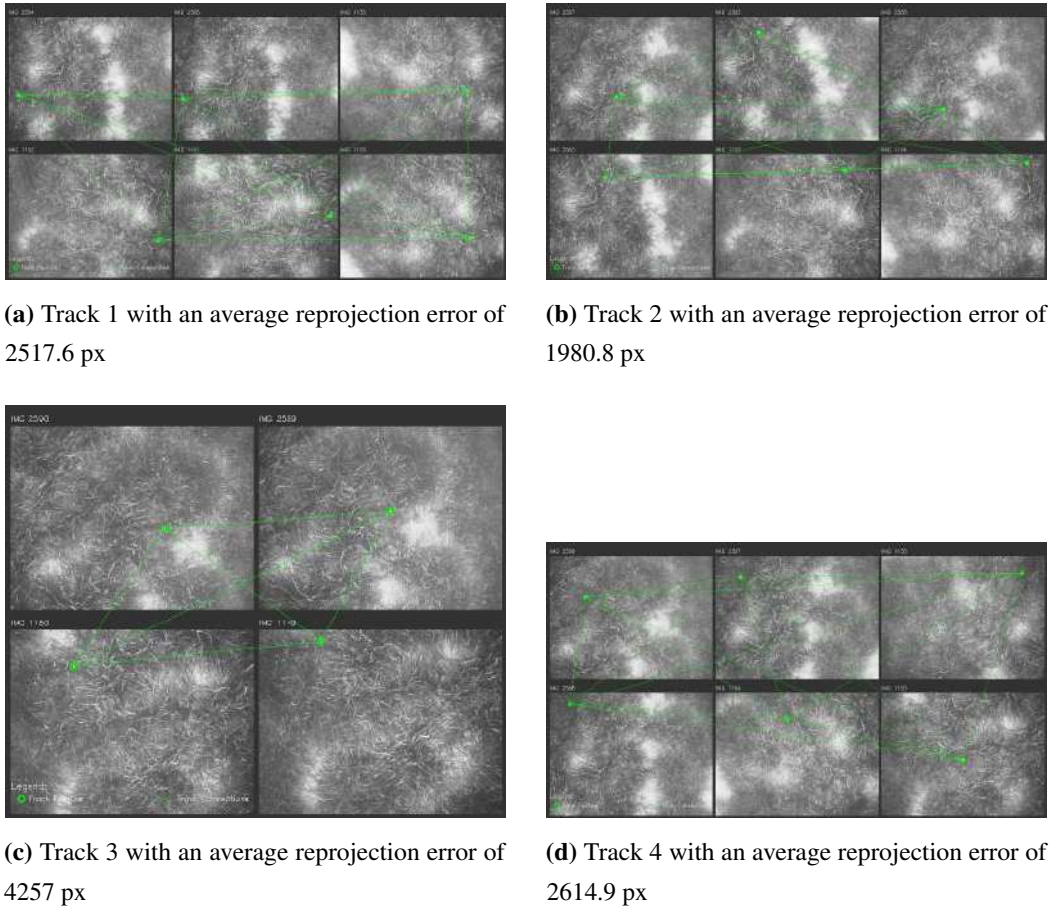


Figure 4.9: Visualization of feature point matching across a grid of grayscale optical images. Keypoints are marked by green dots, with green lines illustrating successful matches.

4.3.3. Qualitative Results

This section presents a qualitative analysis of the final map, focusing on detailed views of specific regions to highlight the improvements in alignment post-optimization. To clearly visualize the corrections, overlays of corresponding sonar and optical data are shown for several key areas.

Figures 4.10 through 4.12 present a comparison of these sonar and optical mosaic patches before and after the optimization. The initial, unoptimized overlays show significant misalignments between features across different sessions and modalities. After optimization, the overlays (highlighted with colored outlines) demonstrate a substantial improvement in the co-registration of corresponding features. This visual evidence confirms the framework’s effectiveness at correcting the large initial offsets and achieving a more coherent, multimodal map.

It is important to analyze the nature of the applied corrections, which are inherently non-rigid. The framework performs a two-level alignment:

- A global rigid transformation via the session anchor, Δ^j , which corrects the overall offset and orientation of each session relative to the world frame.
- A local non-rigid deformation via the optimization of individual state nodes, X_k^j , which allows the trajectory within each session to be warped, stretched, and bent.

This non-rigid capability is designed to address complex, time-varying errors that stem from sources such as accumulated navigation drift, intermittent localization inaccuracies, or imperfect time synchronization of the various sensors.

For instance, Figures 4.11 and 4.14 show a sonar survey with significant trajectory distortions where a purely rigid transformation would not be sufficient. The proposed framework, by allowing local pose adjustments, substantially reduces these nonlinear misalignments. While the optimization does not eliminate all non-rigid effects, the results demonstrate the promise of this approach. Achieving a more perfect correction is a matter of modeling other errors not considered in this work and further tuning the factor weights and noise models, representing a valuable direction for future work.

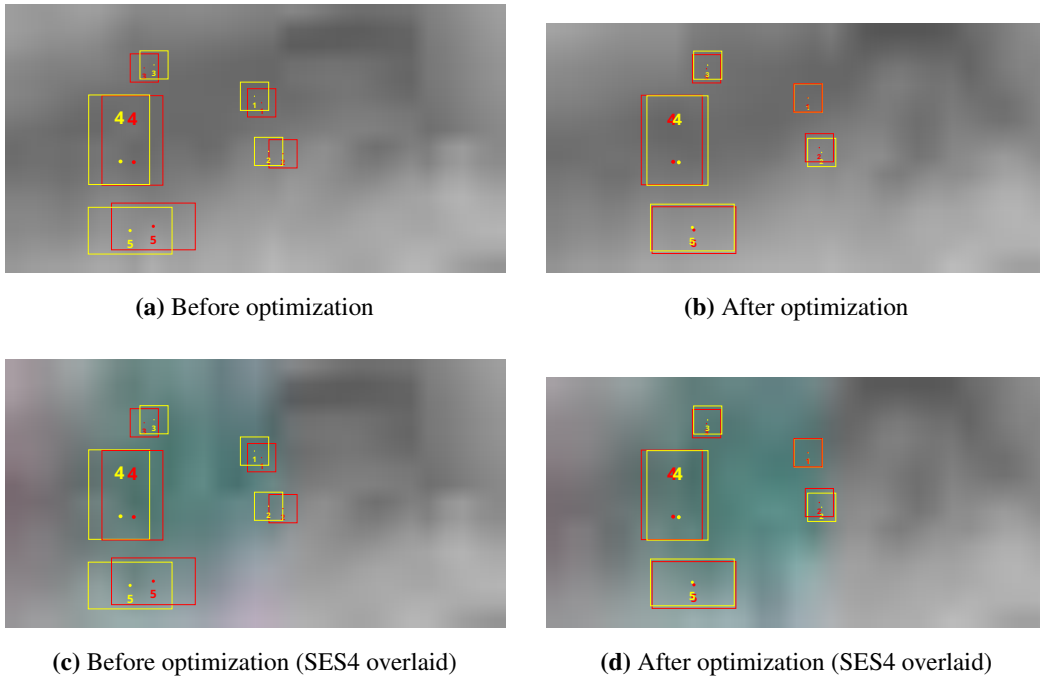


Figure 4.10: Mosaics of SES0, SES2, and SES4 with annotations: red point and box indicate features and area for observation in SES0 & SES2; yellow point and box indicate those for SES4 (optical session).

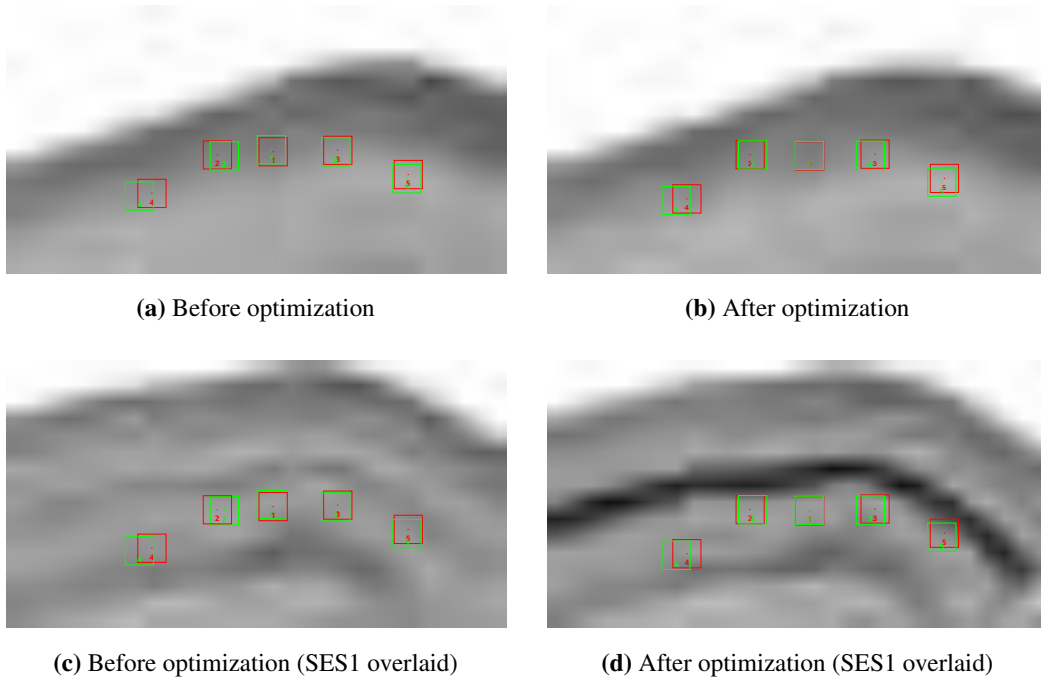


Figure 4.11: Mosaics of SES0 and SES1 with annotations: red point and box indicate features and area for observation in session 0; green point and box indicate those for Session 1.

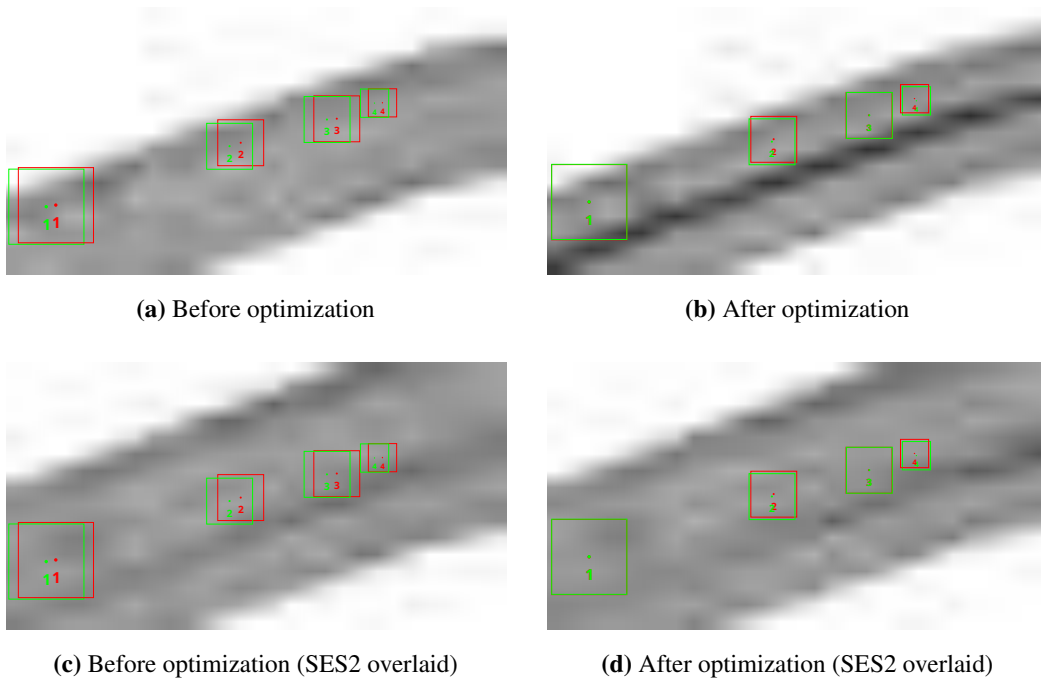


Figure 4.12: Mosaics of SES1 and SES2 with annotations: red point and box indicate features and area for observation in SES1; green point and box indicate those for SES2.

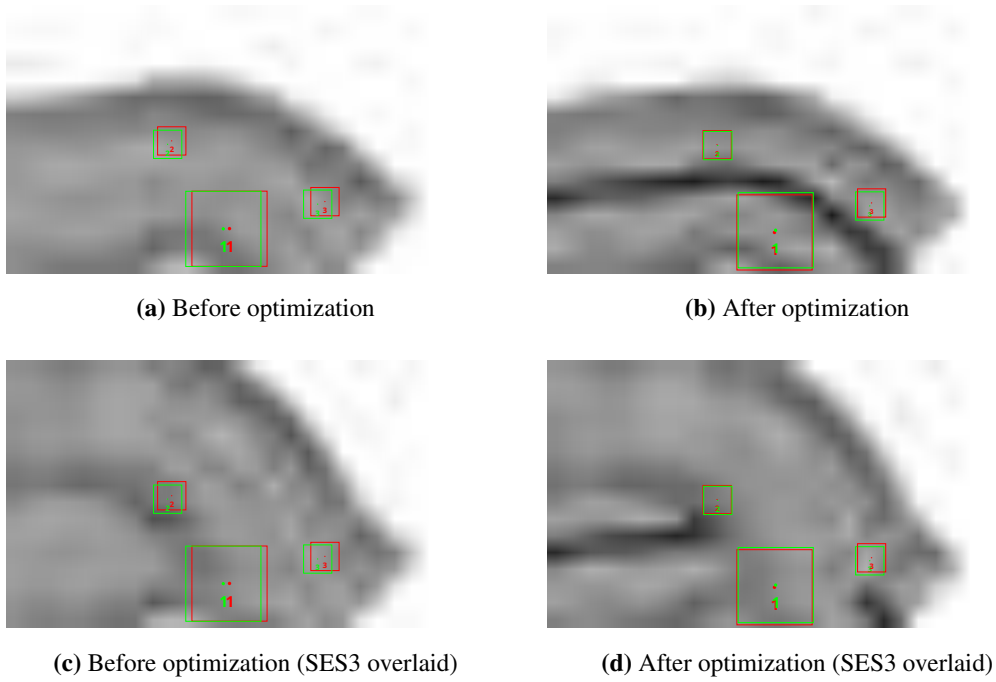


Figure 4.13: Mosaics of SES1 and SES3 with annotations: red point and box indicate features and area for observation in SES1; green point and box indicate those for SES3.

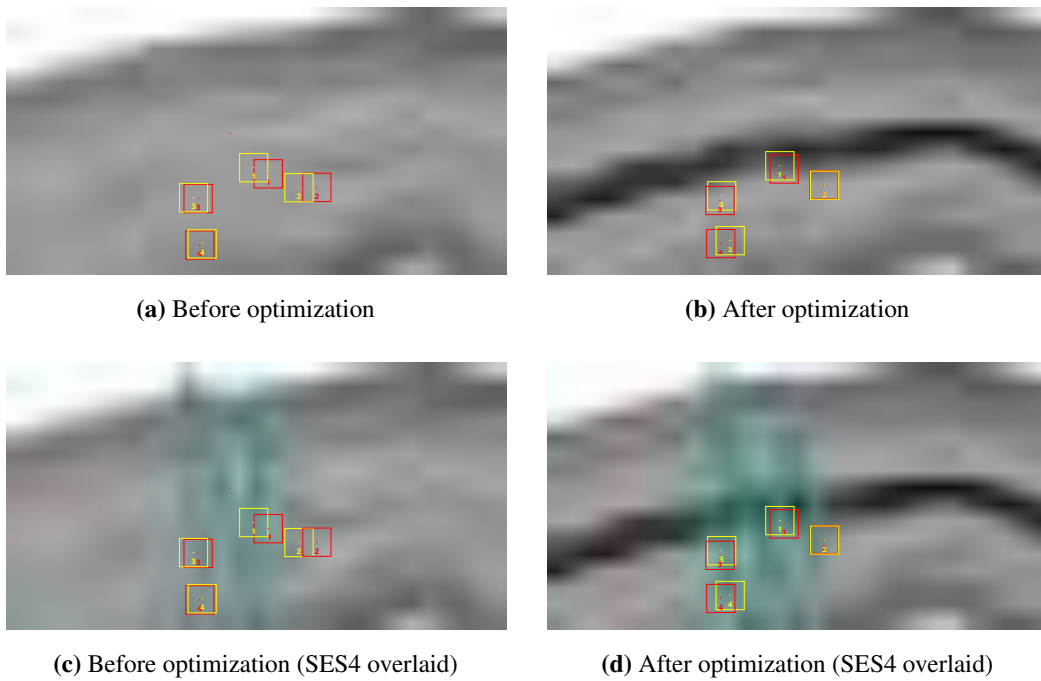


Figure 4.14: Mosaics of SES1 and SES4 with annotations: red point and box indicate features and area for observation in SES1 (sonar session); yellow point and box indicate those for SES4 (optical session).

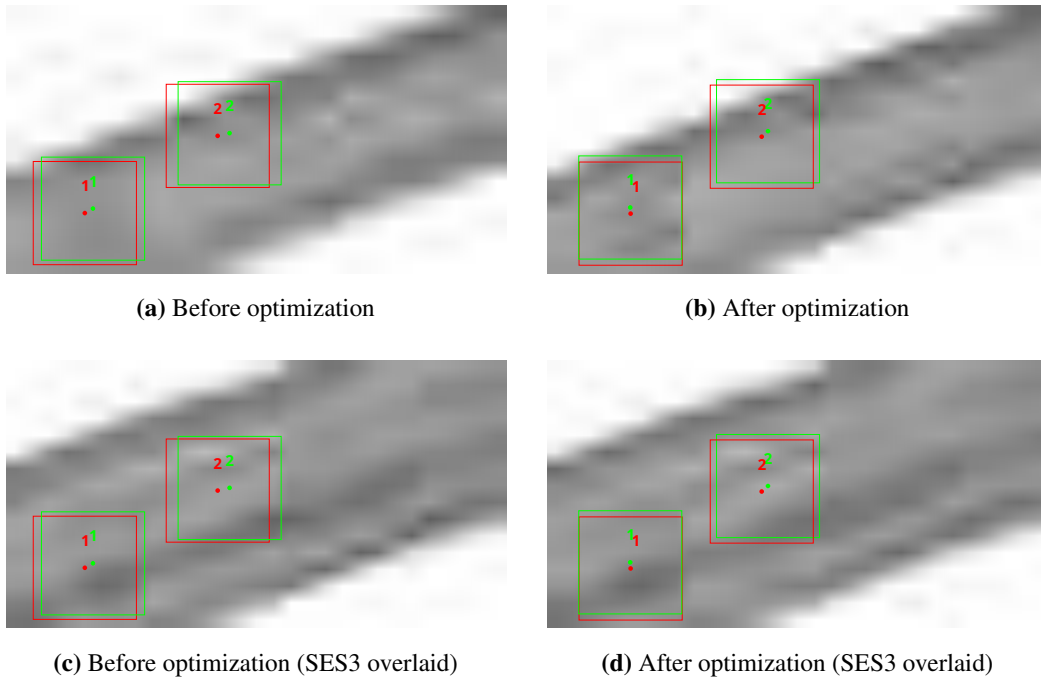


Figure 4.15: Mosaics of SES2 and SES3 with annotations: red point and box indicate features and area for observation in SES2; green point and box indicate those for SES3.

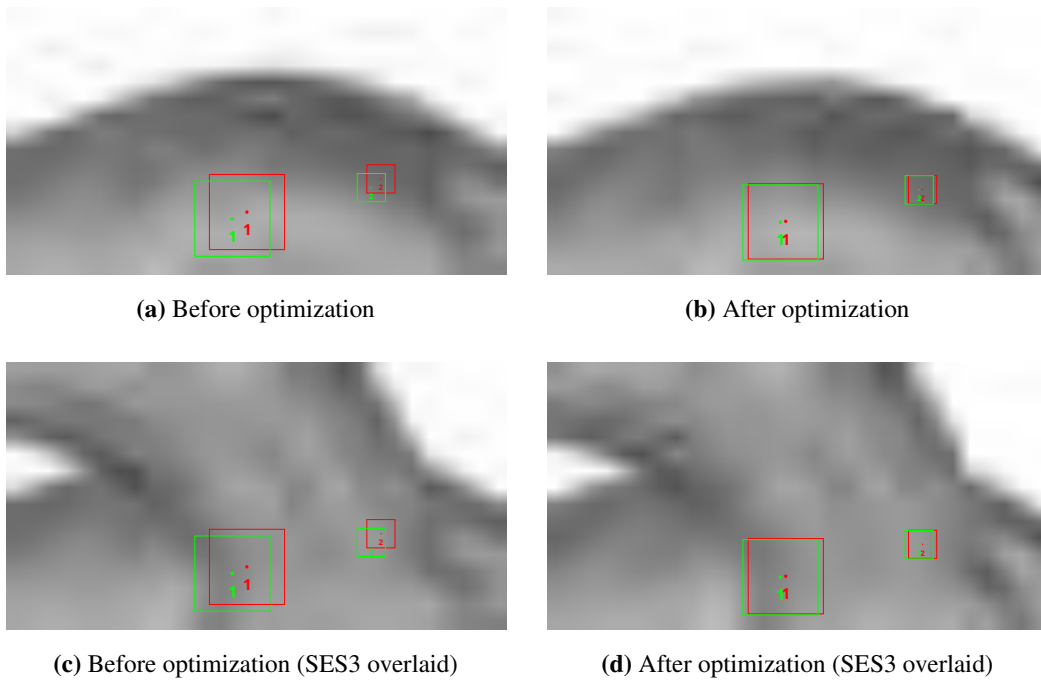


Figure 4.16: Mosaics of SES0 and SES3 with annotations: red point and box indicate features and area for observation in SES0; green point and box indicate those for SES3.

5. Conclusion

This thesis addressed the significant challenge of achieving globally consistent, multi-session, multi-modal underwater mapping by aligning data from side-scan sonar and optical cameras. To solve this, a novel framework based on factor graph optimization was developed. The proposed method performs a comprehensive, non-rigid alignment by jointly optimizing vehicle trajectories, 3D landmark positions, session-to-world transformations, and sensor extrinsics. By using a two-level alignment, a global rigid transformation for each session combined with local non-rigid deformations of the trajectory, the framework is capable of correcting for a wide range of errors, from inter-session offsets to intra-session distortions caused by navigation drift. The experimental results, validated both qualitatively and quantitatively, demonstrated the framework's effectiveness. The optimization significantly reduced projection errors across all sensor modalities. Furthermore, the qualitative results confirmed the correction of large initial misalignments, yielding coherent and accurately co-registered sonar and optical maps. The primary contribution of this work is a unified methodology that can robustly fuse these disparate sensor types across multiple surveys, providing a valuable tool for creating more complete and interpretable models of the seafloor.

5.1. Limitations and Future Work

While this work successfully demonstrates the proposed approach, it also highlights several promising avenues for future research and development.

Automated Cross-Modal Feature Matching The current methodology relies on manual selection of keypoints for sonar-sonar and, most critically, sonar-optical data association. This process can be time-consuming, subjective, and a bottleneck for scaling to larger datasets. A primary direction for future work is the development of automated techniques for this task. This could involve learning-based approaches trained to find robust correspondences between the disparate domains of acoustic

backscatter and optical imagery. This remains a significant research challenge due to the fundamentally different physical principles of the two sensing modalities.

Tightly-Coupled Sensor Fusion The current framework uses pre-processed navigation data (e.g., filtered pose estimates) as an initial guess for the vehicle trajectories. A significant extension would be to create a more tightly-coupled system by incorporating raw sensor measurements from the navigation suite directly into the factor graph.

Long-Term Mapping in Dynamic Environments The datasets used for evaluating this framework were collected years apart, yet the seafloor environment was largely static. A fascinating direction for future work is to extend the framework to handle dynamic environments where significant changes occur between surveys (e.g., sediment movement, seasonal vegetation changes, or new man-made objects). This would involve integrating methods for change detection to distinguish between stable, long-term landmarks and transient features. Such a system would not only align the static parts of the map but also identify and model the changes over time, paving the way for true long-term environmental monitoring.

BIBLIOGRAPHY

- [1] Underwater Vision and Robotics Research Centre at the University of Girona (Vicorob), “IURBI (Intelligent Underwater Robot for Blue Carbon Inventorying) project,” <https://cirs.udg.edu/portfolio/iurbi/>, 2025, Accessed on 2025-06-16.
- [2] M. S. Hitam, E. A. Awalludin, W. N. Jawahir Hj Wan Yussof, and Z. Bachok, “Mixture contrast limited adaptive histogram equalization for underwater image enhancement,” in *2013 International Conference on Computer Applications Technology (ICCAT)*, 2013, pp. 1–5.
- [3] Y. Hao, Y. Yuan, H. Zhang, and Z. Zhang, “Underwater optical imaging: Methods, applications and perspectives,” *Remote Sensing*, vol. 16, no. 20, 2024. [Online]. Available: <https://www.mdpi.com/2072-4292/16/20/3773>
- [4] Y. Song, D. Nakath, M. She, and K. Köser, “Optical imaging and image restoration techniques for deep ocean mapping: a comprehensive survey,” *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 90, no. 3, pp. 243–267, 2022.
- [5] Y. Cong, C. Gu, T. Zhang, and Y. Gao, “Underwater robot sensing technology: A survey,” *Fundamental Research*, vol. 1, no. 3, pp. 337–345, 2021.
- [6] M. K. Moghimi and F. Mohanna, “Real-time underwater image enhancement: a systematic review,” *Journal of Real-Time Image Processing*, vol. 18, no. 5, pp. 1509–1525, 2021.
- [7] S. Raveendran, M. D. Patil, and G. K. Birajdar, “Underwater image enhancement: a comprehensive review, recent trends, challenges and applications,” *Artificial Intelligence Review*, vol. 54, pp. 5413–5467, 2021.
- [8] K. Iqbal, M. Odetayo, A. James, R. A. Salam, and A. Z. H. Talib, “Enhancing the low quality images using unsupervised colour correction method,” in *2010*

- IEEE International conference on systems, man and cybernetics*. IEEE, 2010, pp. 1703–1709.
- [9] S. Sankpal and S. Deshpande, “Underwater image enhancement by rayleigh stretching with adaptive scale parameter and energy correction,” in *Computing, Communication and Signal Processing: Proceedings of ICCASP 2018*. Springer, 2019, pp. 935–947.
 - [10] X. Fu, Z. Fan, M. Ling, Y. Huang, and X. Ding, “Two-step approach for single underwater image enhancement,” in *2017 international symposium on intelligent signal processing and communication systems (ISPACS)*. Ieee, 2017, pp. 789–794.
 - [11] J. S. Jaffe, “Computer modeling and the design of optimal underwater imaging systems,” *IEEE journal of oceanic engineering*, vol. 15, no. 2, pp. 101–111, 1990.
 - [12] B. McGlamery, “A computer model for underwater camera systems,” in *Ocean Optics VI*, vol. 208. SPIE, 1980, pp. 221–231.
 - [13] D. Akkaynak and T. Treibitz, “A revised underwater image formation model,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6723–6732.
 - [14] —, “Sea-thru: A method for removing water from underwater images,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1682–1691.
 - [15] J. Li, K. A. Skinner, R. M. Eustice, and M. Johnson-Roberson, “Watergan: Un-supervised generative network to enable real-time color correction of monocular underwater images,” *IEEE Robotics and Automation letters*, vol. 3, no. 1, pp. 387–394, 2017.
 - [16] C. Li, J. Guo, and C. Guo, “Emerging from water: Underwater image color correction based on weakly supervised color transfer,” *IEEE Signal processing letters*, vol. 25, no. 3, pp. 323–327, 2018.
 - [17] C. Fabbri, M. J. Islam, and J. Sattar, “Enhancing underwater imagery using generative adversarial networks,” in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7159–7165.

- [18] Y. Guo, H. Li, and P. Zhuang, “Underwater image enhancement using a multiscale dense generative adversarial network,” *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, pp. 862–870, 2019.
- [19] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, pp. 91–110, 2004.
- [20] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (surf),” *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [21] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5–11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.
- [22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.
- [23] J. Yang, M. Gong, G. Nair, J. H. Lee, J. Monty, and Y. Pu, “Knowledge distillation for feature extraction in underwater vslam,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5163–5169.
- [24] J. Sun, H. Yu, Y. Rao, and H. Zhang, “Underwater visual feature matching based on attenuation invariance,” *Intelligent Marine Technology and Systems*, vol. 1, no. 1, p. 11, 2023.
- [25] Z. Wang, Q. Cheng, and X. Mu, “Ru-slam: A robust deep-learning visual simultaneous localization and mapping (slam) system for weakly textured underwater environments,” *Sensors*, vol. 24, no. 6, p. 1937, 2024.
- [26] P. Vandrish, A. Vardy, D. Walker, and O. A. Dobre, “Side-scan sonar image registration for auv navigation,” in *2011 IEEE Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies*, 2011, pp. 1–7.
- [27] P. King, B. Anstey, and A. Vardy, “Comparison of feature detection techniques for auv navigation along a trained route,” in *2013 OCEANS - San Diego*, 2013, pp. 1–8.

- [28] K. Kim, N. Neretti, and N. Intrator, "Mosaicing of acoustic camera images," *IEEE Proceedings-Radar, Sonar and Navigation*, vol. 152, no. 4, pp. 263–270, 2005.
- [29] X.-F. Ye, P. Li, J.-G. Zhang, J. Shi, and S.-X. Guo, "A feature-matching method for side-scan sonar images based on nonlinear scale space," *Journal of Marine Science and Technology*, vol. 21, pp. 38–47, 2016.
- [30] J. Zhang, Y. Xie, L. Ling, and J. Folkesson, "A fully-automatic side-scan sonar simultaneous localization and mapping framework," *IET Radar, Sonar & Navigation*, vol. 18, no. 5, pp. 674–683, 2024.
- [31] Y. Yang, C. Pang, C. Wu, and Z. Fang, "Geometry-aided underwater 3d mapping using side-scan sonar," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 1038–1045.
- [32] P. Lindenberger, P.-E. Sarlin, and M. Pollefeys, "LightGlue: Local Feature Matching at Light Speed," in *ICCV*, 2023.
- [33] J. Aulinas, X. Lladó, J. Salvi, and Y. R. Petillot, "Feature based slam using side-scan salient objects," in *OCEANS 2010 MTS/IEEE SEATTLE*. IEEE, 2010, pp. 1–8.
- [34] I. Tena Ruiz, Y. Petillot, and D. Lane, "Improved auv navigation using side-scan sonar," in *Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492)*, vol. 3, 2003, pp. 1261–1268 Vol.3.
- [35] J. Zhang, Y. Xie, L. Ling, and J. Folkesson, "A dense subframe-based slam framework with side-scan sonar," *arXiv preprint arXiv:2312.13802*, 2023.
- [36] E. Westman, A. Hinduja, and M. Kaess, "Feature-based slam for imaging sonar with under-constrained landmarks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3629–3636.
- [37] J. Li, M. Kaess, R. M. Eustice, and M. Johnson-Roberson, "Pose-graph slam using forward-looking sonar," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2330–2337, 2018.
- [38] A. Abu and R. Diamant, "Underwater object classification combining sas and transferred optical-to-sas imagery," *Pattern Recognition*, vol. 144, p. 109868, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0031320323005666>

- [39] H. Jang, G. Kim, Y. Lee, and A. Kim, “Cnn-based approach for opti-acoustic reciprocal feature matching,” in *ICRA Workshop on Underwater Robotics Perception*, vol. 1, 2019.
- [40] X. Zhou, C. Yu, X. Yuan, and C. Luo, “A matching algorithm for underwater acoustic and optical images based on image attribute transfer and local features,” *Sensors*, vol. 21, no. 21, 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/21/7043>
- [41] H. Jang, S. Yoon, and A. Kim, “Multi-session underwater pose-graph slam using inter-session opti-acoustic two-view factor,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 668–11 674.
- [42] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “SuperGlue: Learning feature matching with graph neural networks,” in *CVPR*, 2020.
- [43] J. Liu, B. Li, W. Guan, S. Gong, J. Liu, and J. Cui, “A scale-adaptive matching algorithm for underwater acoustic and optical images,” *Sensors*, vol. 20, no. 15, p. 4226, 2020.
- [44] X. Jiang, J. Ren, Z. Li, X. Zhou, D. Liang, and X. Bai, “Minima: Modality invariant image matching,” *arXiv preprint arXiv:2412.19412*, 2024.
- [45] X. He, H. Yu, S. Peng, D. Tan, Z. Shen, H. Bao, and X. Zhou, “Matchanything: Universal cross-modality image matching with large-scale pre-training,” in *Arxiv*, 2025.
- [46] A. Cardaillac and M. Ludvigsen, “Camera-sonar combination for improved underwater localization and mapping,” *IEEE Access*, vol. 11, pp. 123 070–123 079, 2023.
- [47] H. Qiu, Y. Tang, H. Wang, L. Wang, D. Xiang, and M. Xiao, “An improved underwater visual slam through image enhancement and sonar fusion,” *Remote Sensing*, vol. 16, no. 14, p. 2512, 2024.
- [48] J. Zhang, F. Han, D. Han, J. Yang, W. Zhao, and H. Li, “Integration of sonar and visual inertial systems for slam in underwater environments,” *IEEE Sensors Journal*, 2024.

- [49] F. Ferreira, D. Machado, G. Ferri, S. Dugelay, and J. Potter, “Underwater optical and acoustic imaging: A time for fusion? a brief overview of the state-of-the-art,” in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1–6.
- [50] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard, “Advanced perception, navigation and planning for autonomous in-water ship hull inspection,” *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1445–1464, 2012.
- [51] C. G. Kunz, “Autonomous underwater vehicle navigation and mapping in dynamic, unstructured environments,” Ph.D. dissertation, Massachusetts Institute of Technology, 2012.
- [52] S. Negahdaripour, H. Sekkati, and H. Pirsiavash, “Opti-acoustic stereo imaging: On system calibration and 3-d target reconstruction,” *IEEE Transactions on image processing*, vol. 18, no. 6, pp. 1203–1214, 2009.
- [53] M. Kaess, A. Ranganathan, and F. Dellaert, “isam: Incremental smoothing and mapping,” *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.
- [54] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, “Multiple relative pose graphs for robust cooperative mapping,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3185–3192.
- [55] S. Rahman, A. Quattrini Li, and I. Rekleitis, “Svin2: A multi-sensor fusion-based underwater slam system,” *The International Journal of Robotics Research*, vol. 41, no. 11-12, pp. 1022–1042, 2022.
- [56] A. Burguera and F. Bonin-Font, “On-line multi-class segmentation of side-scan sonar imagery using an autonomous underwater vehicle,” *Journal of Marine Science and Engineering*, vol. 8, no. 8, p. 557, 2020.
- [57] P. Blondel, *The handbook of sidescan sonar*. Springer Science & Business Media, 2010.
- [58] A. Burguera and G. Oliver, “High-resolution underwater mapping using side-scan sonar,” *PloS one*, vol. 11, no. 1, p. e0146396, 2016.
- [59] O. Leenhardt, “Side scanning sonar-a theoretical study,” *The International Hydrographic Review*, 1974.

- [60] P. Woock and C. Frey, “Deep-sea auv navigation using side-scan sonar images and slam,” in *OCEANS’10 IEEE SYDNEY*. IEEE, 2010, pp. 1–8.
- [61] J. S. Jaffe, “Underwater optical imaging: the past, the present, and the prospects,” *IEEE Journal of Oceanic Engineering*, vol. 40, no. 3, pp. 683–700, 2014.
- [62] A. Barcelona, J. Colomer, M. Soler, N. Gracias, and T. Serra, “Meadow fragmentation influences posidonia oceanica density at the edge of nearby gaps,” *Estuarine, Coastal and Shelf Science*, vol. 249, p. 107106, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0272771420308374>
- [63] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, “Real-time 6-dof multi-session visual slam over large-scale environments,” *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1144–1158, 2013.
- [64] F. Dellaert and G. Contributors, “borglab/gtsam,” May 2022. [Online]. Available: <https://github.com/borglab/gtsam>)
- [65] J. Escartín, R. Garcia, O. Delaunoy, J. Ferrer, N. Gracias, A. Elibol, X. Cufi, L. Neumann, D. Fornari, S. Humphris *et al.*, “Globally aligned photomosaic of the lucky strike hydrothermal vent field (mid-atlantic ridge, 37 18.5’ n): Release of georeferenced data, mosaic construction, and viewing software,” *Geochemistry, Geophysics, Geosystems*, vol. 9, no. 12, 2008.
- [66] D. W. Caress and D. N. Chayes, “MB-System,” <https://www.mbari.org/technology/mb-system>, 2024, references to the software distributions.
- [67] I. Klein Marine Systems, “Klein system 3000 dual-frequency side scan sonar brochure,” https://kordil.com/application/files/2815/9301/6165/Klein_System_3000.pdf, 2025, accessed: 2025-06-01.

Joint Underwater Mapping with Acoustic and Optical Images

Sažetak

Točno i sveobuhvatno mapiranje morskog dna ključno je za brojne primjene u znanosti o moru, arheologiji i praćenju okoliša. Međutim, integriranje podataka s različitih senzora, poput bočnog sonara i optičkih kamera, prikupljenih tijekom više sesija istraživanja, predstavlja značajan izazov zbog navigacijskih netočnosti, pomaka senzora i vremenski promjenjivih pogrešaka poput pomicanja.

Ova teza predstavlja novi okvir za višesesijsko, multimodalno podvodno mapiranje koji rješava ovaj izazov. Problem formuliramo korištenjem optimizacije faktorskog grafa za izvođenje sveobuhvatnog, nerigidnog poravnanja. Metoda zajednički optimizira sve ključne varijable: putanje vozila, položaje 3D orijentira, vanjske značajke senzora i globalne transformacije poravnanja (sidra) po sesiji. Ovaj dvorazinski pristup, koji kombinira globalnu krutu korekciju za svaku sesiju s lokalnim nerigidnim deformacijama putanje, omogućuje okviru da ispravi širok raspon geometrijskih nedosljednosti.

Metodologija je validirana na složenom skupu podataka iz stvarnog svijeta koji se sastoji od četiri sonara i jedne optičke sesije prikupljene duž katalonske obale. Kvantitativna analiza pokazuje značajno smanjenje pogrešaka u projekciji, potvrđujući prednosti multimodalnog fuzijskog pristupa. Kvalitativno, okvir uspješno ispravlja početne neusklađenosti, dajući koherentne i točno koregistrirane sonarne i optičke karte, čak i uz prisutnost značajnih izobličenja putanje unutar sesije.

Primarni doprinos ovog rada je ujedinjena i robusna metodologija za spajanje različitih podataka podvodnih senzora u više kampanja. Učinkovitim kombiniranjem širokog područja pokrivenosti sonara i detalja optičkih slika visoke rezolucije, ovaj okvir pruža moćan alat za stvaranje potpunijih, točnijih i interpretabilnijih modela morskog okoliša.

Ključne riječi: Kartiranje morskog dna, multimodalna fuzija podataka, podvodna robotika

Abstract

Accurate and comprehensive mapping of the seafloor is crucial for numerous applications in marine science, archaeology, and environmental monitoring. However, integrating data from different sensors, such as side-scan sonar and optical cameras, collected across multiple survey sessions presents a significant challenge due to navigational inaccuracies, sensor offsets, and time-varying errors like drift.

This thesis presents a novel framework for multi-session, multi-modal underwater mapping that addresses this challenge. We formulate the problem using factor graph optimization to perform a comprehensive, non-rigid alignment. The method jointly optimizes all key variables: vehicle trajectories, 3D landmark positions, sensor extrinsics, and per-session global alignment transformations (anchors). This two-level approach, combining a global rigid correction for each session with local non-rigid deformations of the trajectory, allows the framework to correct for a wide range of geometric inconsistencies.

The methodology was validated on a complex real-world dataset comprising four sonar and one optical session collected along the Catalan coast. Quantitative analysis demonstrates a significant reduction in projection errors, validating the benefits of the multimodal fusion approach. Qualitatively, the framework successfully corrects initial misalignments, yielding coherent and accurately co-registered sonar and optical maps, even in the presence of significant intra-session trajectory distortions.

The primary contribution of this work is a unified and robust methodology for fusing disparate underwater sensor data across multiple campaigns. By effectively combining the wide-area coverage of sonar and the high-resolution detail of optical imagery, this framework provides a powerful tool for creating more complete, accurate, and interpretable models of the marine environment.

Keywords: Seafloor Mapping, Multi-modal Data Fusion, Underwater Robotics

18

**Diurnal and Nocturnal Robust Visual-Aided GNSS Navigation on
Horticultural Fields**

Lisa Paul Magoti



Eötvös Loránd University

Faculty of Informatics

Erasmus Mundus Joint Master in
Intelligent Field Robotic Systems

Diurnal and Nocturnal Robust Visual-Aided GNSS Navigation on Horticultural Fields

Author:

Lisa Paul Magoti

EMJM Intelligent Field Robotic Systems

Internal supervisor:

Dr. Zoltán Istenes

Professor, ELTE

External supervisor:

Bram Benist

Senior Robotics Engineer, Agrikola.AI

Budapest, 2025

Abstract

The pursuit of increased efficiency, sustainability, and reduced labor dependency in agriculture has driven significant advancements in robotic autonomy for horticultural fields. Autonomous robots can perform repetitive and labor-intensive tasks such as weeding, harvesting, and monitoring with high precision, supporting more sustainable and productive farming practices. However, achieving reliable autonomy remains challenging due to environmental variability, inconsistent terrain, and the structural complexity of crop arrangements. A particularly critical issue is the misalignment between predefined crop bed layouts based on GNSS data and the actual physical positions of beds in the field, deviations that can compromise navigation accuracy and crop safety. This thesis addresses this challenge by enabling agricultural robots to perceive and adapt to the true structure of horticultural beds through real-time visual input, rather than relying solely on preconfigured positional data. By focusing on accurate crop row detection and navigation, the proposed approach enhances the robot's ability to operate robustly in dynamic field conditions, paving the way for more precise and autonomous horticultural systems.