

## **Final Project II**

# **Collective Transport using Decentralised Swarm Robotics**



**Submitted to the**  
Project Committee appointed by the  
**International School of Engineering (ISE)**  
Faculty of Engineering, Chulalongkorn University

**Project Adviser**  
Asst.Prof.Paulo Fernando Rocha Garcia, Ph.D.

### **Submitted By**

6438067021	Nattadon Tangsasom
6438075021	Ting-Yi Lin
6438079621	Tinapat Limsila
6438118421	Noppawan Srihirin
6438187721	Mehul Sharma

1/2025: 2147417 Final Project II  
Robotics and Artificial Intelligence Engineering (International Programme)  
International School of Engineering (ISE) Faculty of Engineering, Chulalongkorn  
University

# Contents

<b>1</b>	<b>Introduction and Overview</b>	<b>2</b>
<b>2</b>	<b>Literature Survey and Review</b>	<b>9</b>
2.1	Coordination . . . . .	9
2.2	Object Detection . . . . .	10
2.3	SLAM . . . . .	12
2.4	Collective Movement . . . . .	15
<b>3</b>	<b>Hardware</b>	<b>18</b>
3.1	Hardware Layout . . . . .	19
3.2	Robot Frame . . . . .	20
3.3	Robot Drive Train and kinematic . . . . .	22
3.4	Base Platform Modification and Microcontroller Integration . . . . .	24
<b>4</b>	<b>Object Detection Using Computer Vision</b>	<b>26</b>
<b>5</b>	<b>Communication in the Swarm</b>	<b>33</b>
<b>6</b>	<b>Coordination and Formation</b>	<b>40</b>
<b>7</b>	<b>Localization</b>	<b>45</b>
<b>8</b>	<b>Result</b>	<b>53</b>
<b>9</b>	<b>Conclusion</b>	<b>57</b>
<b>A</b>	<b>Object Detection Model Testing Result</b>	<b>59</b>

# Abstract

The rapid advancements in robotics have created significant opportunities to address complex tasks through collaborative systems. This report explores the project "**Collective Transport using Decentralised Swarm Robotics**". The core problem addressed is the difficulty of achieving effective collaboration between decentralised robots, particularly in tasks such as collectively transporting objects in unstructured environments. Unlike centralised systems, which are vulnerable to single-point failures, decentralised swarm systems offer robust, scalable, and efficient solutions.

This document provides an in-depth analysis of key components, including communication protocols, object detection, localisation, SLAM, and hardware integration. It also includes a comprehensive literature review and a solid theoretical framework underpinning the project's objectives. The report is organised into nine chapters, beginning with an introduction and a comprehensive literature review, followed by an overview of the simulation. Subsequent chapters delve into the primary aspects of the project: communication, object detection, localisation, SLAM, and robot formation.

This report summarises the progress achieved during this semester and provides a comprehensive overview of our work on the project.

# Chapter 1

## Introduction and Overview

The project, “Collective Transport using Decentralised Swarm Robotics,” aims to pioneer a new approach to robotic collaboration by enabling decentralised control for tasks such as object transportation in unstructured environments. Traditional centralised systems, like those in RoboCup Soccer, have dominated the field for decades but face limitations such as single points of failure and reliance on powerful centralised servers. This project seeks to address these challenges by leveraging decentralised swarm robotics, which ensures robust, scalable, and efficient operations without dependency on a single control unit.

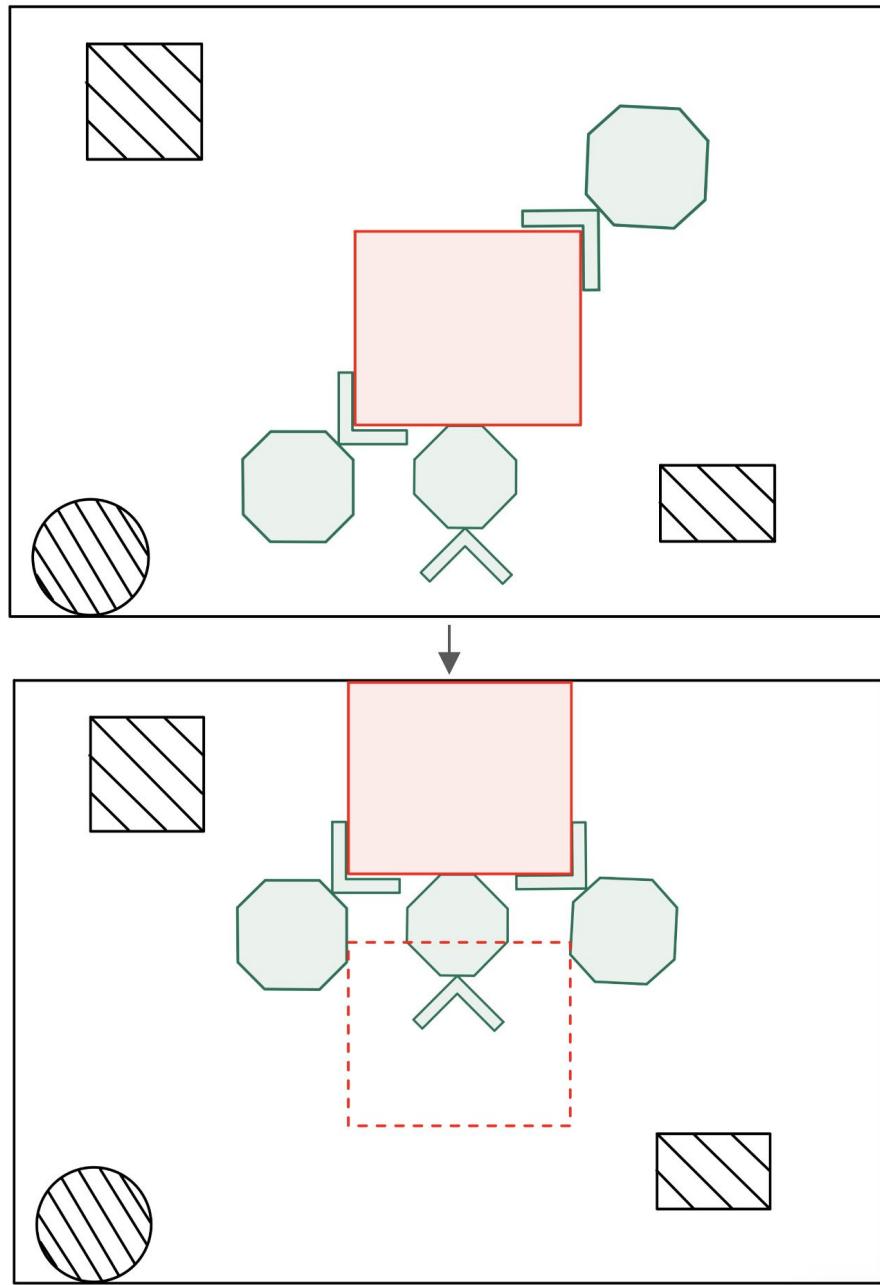


Figure 1.1: Illustration of the project's goal: three robots transporting an object using a drawing

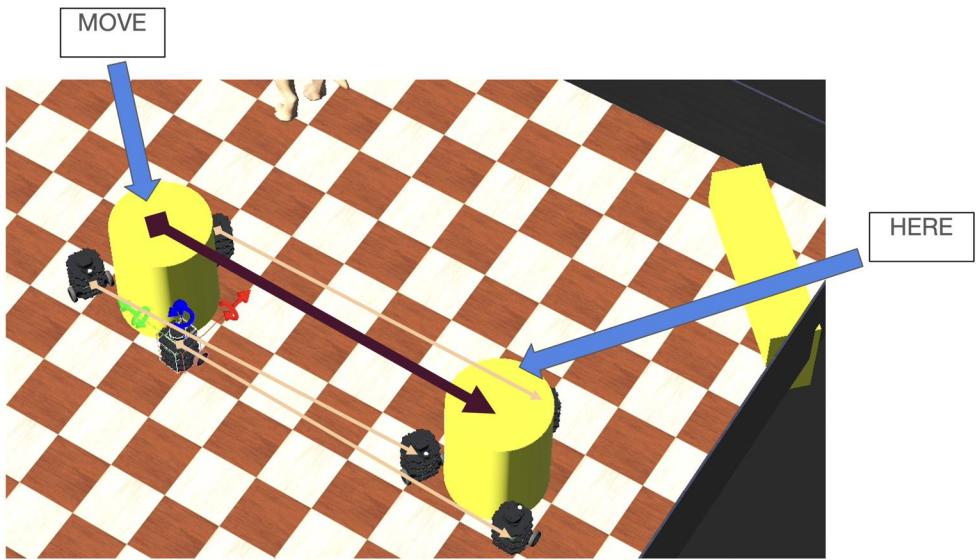


Figure 1.2: Illustration of the project's goal: three robots transporting an object using simulation

A core objective of the project is to develop a system where three robots can simultaneously map their environment, communicate with one another, and collaborate to transport an object from point A to point B in a synchronised manner. This involves the robots autonomously detecting an object, localising themselves in the environment, and coordinating their movements to ensure seamless and efficient transport. This goal demonstrates the potential of decentralised swarm robotics for highly coordinated tasks that require precise communication and execution. See Figure 1.2 for a simulation of the project's goal using the simulation. See Figure 1.1 for an illustration of the project's goal using a drawing

In real-world applications, this approach holds transformative potential, particularly in warehouse logistics and search-and-rescue operations. The decentralised model offers resilience and adaptability, making it suitable for scenarios where centralised systems may falter. Moreover, this technology aligns with Thailand's growing need for innovative solutions in robotics, showcasing a shift towards decentralised systems that can operate independently without a centralised power hub. This represents a vision for the future of robotics, where hardware and computation costs decrease, enabling accessible and scalable robotic systems.

The project development is divided into two phases:

1. Phase 1: Exploration of foundational modules, including communication protocols, object detection via computer vision, and localisation. While the original plan included SLAM, it was deprioritised due to its limited necessity in the simulation phase. A significant focus was placed on ensuring the robustness of these individual modules, validated through simulations in Webots, which provided a cost-efficient, risk-free environment.

2. Phase 2: Integration of these foundational modules into a working prototype of robots navigating their environment. This phase also incorporates initial hardware configurations and an overhead camera to provide accurate odometry information for the robots. The ultimate goal is to transition from simulation to real-world applications, with the simulation results guiding the hardware design.

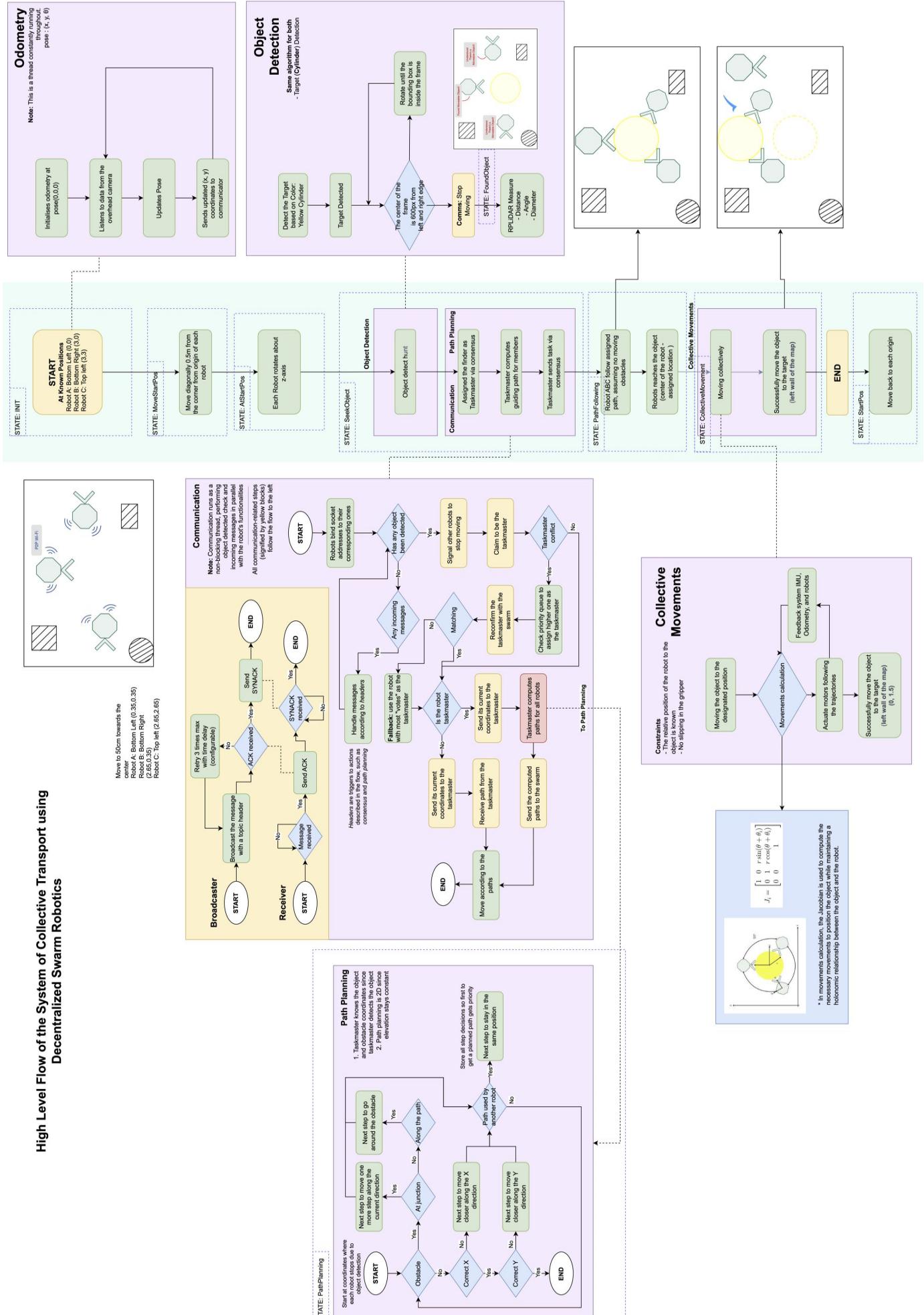


Figure 1.3: High Level Flow Diagram of the project

The high level flowchart above is a detailed look at how different modules of our project come together. More details on each module will be expanded on later.

The success of the project is measured not only by the accuracy and control of individual robots but also by the robustness of their communication and coordination. Metrics include the ability to resolve edge conditions, such as simultaneous object detection or potential collision paths, and the effectiveness of decentralised decision-making. Overhead cameras verify the robots' positional accuracy, ensuring that simulated and real-world performance align.

This report highlights the project's accomplishments, including completed tasks like communication, object detection, and localisation, alongside the progress of module integration. It also underscores the importance of simulation as a foundation for hardware development and presents a roadmap for achieving decentralised, collaborative transport systems. Through this effort, the project aims to establish a framework that not only advances robotics technology but also addresses challenges specific to Thailand, paving the way for future innovations in swarm robotics.

Building on the progress achieved in the previous semester, we successfully developed a Minimum Viable Prototype (MVP) of a swarm collective movement system within a simulated environment. This accomplishment sets the foundation for the current semester, where our initial focus is on preparing for the hardware implementation phase. Subsequently, we aim to realize our proposed concept through deployment on a physical swarm system, transitioning this towards the real-world implementation.

This final report aims to highlight the progress of our project throughout the whole semester, with the evaluating criteria being the team's contributions alongside the pace in comparison to the ideal schedule. The ideal project timeline is presented with the Gantt Chart below. (Figure 1.4)

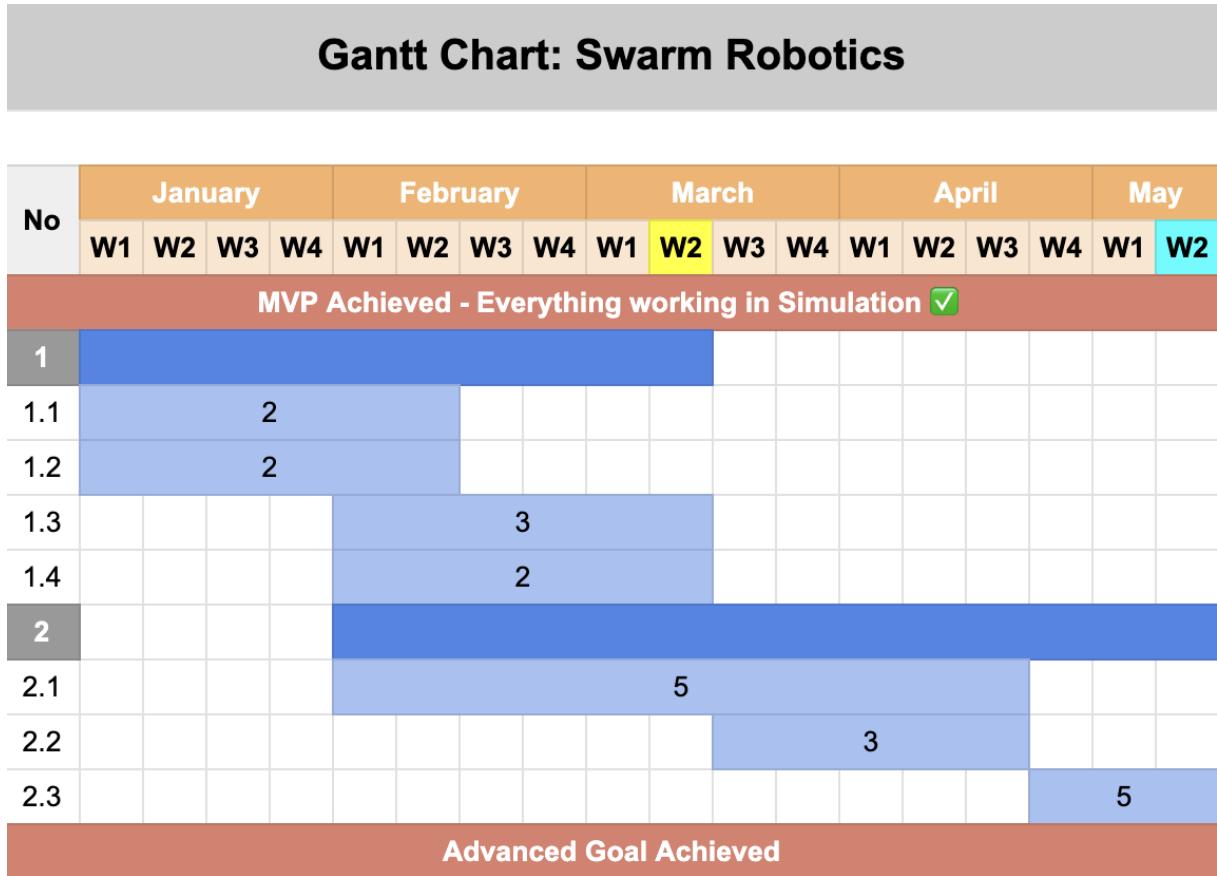


Figure 1.4: Project Gantt Chart

### Current Gantt Chart and Progress:

#### Phase 1: Preparation for Hardware Implementation

- 1.1. Communication in the Swarm – *Completed* – Achieved communication in the swarm by using socket communication in a peer-to-peer architecture, alongside a consensus algorithm to handle communicatory conflicts
- 1.2. SLAM in Simulation – *Status* – Placeholder for details
- 1.3. Coordination and Formation – *Completed* – Enabled collisionless, planned movement towards a coordinated formation using waypoints, forming an equilateral formation around the detected target
- 1.4. Object Detection – *Completed* – Measure relative distance, relative angle, and diameter of the detected cylinder using multi-sensors fusion; camera and S3 RPLiDAR

#### Phase 2: Moving Towards a Complete Swarm

- 2.1. Hardware – *Status* – Placeholder for details
- 2.2. Movement after Gripping – *Aborted* – Placeholder for details
- 2.3. Testing and Evaluation – *Status* – Placeholder for details

# Chapter 2

## Literature Survey and Review

### 2.1 Coordination

Swarm robotics solution is a design architecture that obtains inspirations from biological interactions; thus, they should operate autonomously to solve problems rather than relying on a central authority[1]. This decentralised approach is also crucial to achieve increased resilience and flexibility given a complex environment of deployment[2].

Communication protocols can be categorised into two principal types depending on the nature of information transmission: Direct communication and Indirect communication[2]. Direct communication refers to robotic agents that are able to coordinate via networked communication. Direct communication use cases are highlighted in many studies.

Ibrahim et al. [3] studied direct communication to determine the optimal distance for robotic agents to achieve consensus. Three strategies were tested within a 50 cm range: Close-neighbour, Far-neighbour, and Rand-neighbour. Close-neighbour excels in stable environments but performs poorly in complex ones. Both Close-neighbour and Far-neighbour introduce bias, reducing accuracy. Rand-neighbour, which randomly selects swarm members for communication, proved superior due to efficient information flow and minimal bias. This strategy can be one of our key designs for swarm communication in this project.

Ayari and Bouamama[4] and Perera et al.[5], S et al.[6] and Z. Wang et al.[7] conducted studies to promote robots' actions based on their sensory observations and objectives. S et al.[6] proposed a Multi-Agent Deep Deterministic Policy Gradient (MADDPG), an observation-based decision making flow with an architectural twist, where there is a central swarm manager that evaluates decentralized agents' reinforcement learning for optimal rewards. Z. Wang et al.[7] proposed increasing sensory inputs from both visual data and communication for redundancy, which highly aligns with the proposed swarm system.

Yasser et al.[8] expanded more on Clustered Dynamic Task Allocation (CDTA), an approach to dynamically assign tasks based on swarm state and the environment [9], for the purpose of increasing the velocity of swarm communication. They proposed

CDTA-CL (Centralized Loop) and CDTA-DL (Dual Loop). CDTA-CL sends information to the leader for computation, while CDTA-DL compares information before sending it to the leader. CDTA-DL outperformed CDTA-CL, increasing speed by 75.976% compared to 54.4%. CDTA-DL will be considered as a design pillar for this project.

In addition to direct communication improvements, indirect communication, known as Stigmergy, also plays a significant role in swarm intelligence. This concept involves individual robot actions modifying the environment, impacting the decision-making of other robots. For example, construction robots can leave blocks and materials to signal ongoing work[2]. This is an intriguing notion to consider in the interaction of the swarm system.

For effective communication, especially in interdependent tasks, robots need to be aware of each other and task requirements. Semantic communication, where contextually relevant data is prioritized, is necessary[10]. The balance between communication and context must align with task demands. High sensory data tasks require reliable, real-time communication, while tasks with lower communication demands can prioritize contextually relevant information[11].

## 2.2 Object Detection

Object detection is an essential component of this project, enabling each member of the swarm to perceive their environment and effectively interact with various targets. Traditional 2D object detection methods face challenges in dynamic and complex settings, where accurate estimation of size, distance, and position is critical [12]. To address these issues, integrating a camera sensor with a LiDAR system offers a reliable solution for 3D object detection [13], enabling precise measurement of the distance to the target as well as its dimensions.

Object detection algorithms are typically classified into three categories based on the sensors utilised: camera-based algorithms, LiDAR-based algorithms, and LiDAR-camera fusion algorithms. The first category involves the use of a monocular camera for detection. Object detection with monocular cameras can be achieved through either traditional methods or deep learning-based approaches.

Traditional methods, such as the Scale-Invariant Feature Transform (SIFT), identify key points in images that are invariant to rotation and scaling, making them suitable for recognising objects under various transformations [14]. Another widely used traditional approach is the Histogram of Oriented Gradients (HOG), which captures the distribution of gradient orientations within an image [15]. Both SIFT and HOG have been instrumental in earlier object detection tasks due to their effectiveness in identifying salient features.

In contrast, deep learning-based approaches have revolutionised object detection by surpassing traditional methods in terms of accuracy and adaptability. Prominent algorithms such as YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), and Faster R-CNN offer unique advantages and trade-offs [16]. YOLO is renowned for its real-time processing capabilities, making it well-suited for applications requiring

high-speed detection without significant compromises in accuracy [17]. Faster R-CNN employs a two-stage process that prioritises detection accuracy, although at the cost of slower inference speeds, making it ideal for tasks where precision is critical [18]. Meanwhile, SSD strikes a balance between speed and accuracy by using a single-shot detection mechanism with moderate computational requirements [19].

A comparison of these three models is shown in Table 2.1, focusing on their speed, precision, and adaptability. The evaluation results demonstrate that YOLOv8 excels in multi-object detection, combining high accuracy with balanced speed and efficiency. This makes it a preferred choice for real-time applications due to its lower hardware demands [20].

Algorithms	Recall Value	Mean Average Precision (mAP@0.5)	Precision
YOLOv8	1.00	98.7%	96.77%
SSD	0.65	77%	89%
Faster R-CNN	0.67	59%	77%

Table 2.1: A comparative analysis of YOLOv8, SSD, and Faster R-CNN based on key performance metrics, including Recall Value, Mean Average Precision (mAP@0.5), and Precision [21].

Monocular camera-based object detection relies solely on 2D image data to identify objects and approximate their depth and location using visual cues. While such cameras are effective in detecting colour and texture, estimating depth accurately often requires stereo vision or additional sensors.

LiDAR-based object detection algorithms provide precise spatial and depth information in the form of 3D point clouds, making them highly effective for localising objects in complex environments, particularly under poor lighting conditions where cameras may struggle [22]. LiDAR-based approaches can be broadly categorised into point-based and voxel-based methods.

Point-based methods operate directly on raw 3D point clouds, leveraging their inherent spatial accuracy without converting the data into grids or images. For example, PointNet employs a shared Multi-Layer Perceptron (MLP) to extract features from individual points and aggregates these using max-pooling to capture global context [23]. PointNet++ enhances this approach by introducing hierarchical feature extraction to capture local geometric features, enabling better performance in cluttered or large-scale environments [24].

Conversely, voxel-based methods transform raw point clouds into structured voxel grids, allowing for efficient feature extraction using 3D convolutional neural networks (CNNs). A notable example is VoxelNet, which divides the 3D space into voxels and extracts features using PointNet-inspired architectures, followed by 3D CNNs for detection [25]. Advances such as SECOND improve computational efficiency by optimising voxel representations and employing sparse convolutions [26].

While LiDAR provides unparalleled depth and spatial accuracy, it lacks the rich texture and colour information necessary for tasks such as object classification and semantic segmentation. LiDAR-camera fusion addresses these limitations by combining the strengths of both sensors. Models such as Frustum PointNet project 2D image proposals onto LiDAR point clouds, improving object localisation and classification [27]. Frameworks like MV3D and AVOD demonstrate how fusing RGB images and LiDAR data achieves significant performance improvements [28].

The integration strategy for LiDAR-camera fusion is crucial and can be classified into three categories:

- **Early Fusion:** Aligns raw LiDAR point clouds with camera frames for unified processing [28].
- **Mid-Level Fusion:** Merges features extracted separately from LiDAR and camera data [29].
- **Late Fusion:** Combines decisions made by independent LiDAR and camera models [27].

Another critical task in object detection is determining the position and orientation of objects in the environment. Accurate pose estimation enables robots to interact effectively with their surroundings for tasks such as navigation, manipulation, and object placement [30]. Pose estimation methods are broadly categorised into model-based approaches, such as Iterative Closest Point (ICP) [31], and learning-based methods, including PoseNet [32] and PVNet [33].

Common benchmarks for evaluating pose estimation algorithms include datasets such as LINEMOD and YCB-Video, which offer annotated images and depth maps under varying poses and lighting conditions. However, challenges such as dynamic environments, real-time constraints, and robust performance under occlusions remain areas for further research [34].

## 2.3 SLAM

SLAM, or Simultaneous Localization and Mapping, is a widely spread algorithm for navigation in the field of mobile robotics because of the exponential improvement in computer processing speed and the accessibility of sensors such as cameras and LiDAR [35]. Using SLAM, a mobile robot can construct an internal environment map while simultaneously using the map to estimate its location without needing predefined knowledge of area [36].

Environment mapping is one of the vital techniques in SLAM. The algorithm consists of building a mathematical model for the spatial information of an actual environment, which encapsulates the necessary information for navigation and interaction. However, as for the SLAM technique, additional requirements are needed; the mathematical model must be able to represent the robot's state and the position of landmarks relative to the robot's location [36]. Hence, the challenge with the requirements is that the robot must perform the localization and the mapping simultaneously.

Given these complexities, the backbone of all principal SLAM methods is the utilization of these SLAM frameworks consisting of odometry, landmark prediction, landmark prediction, landmark extraction, data association, and matching, pose estimation, and map update [37].

Building on this, situational awareness becomes an extreme component of SLAM, the precision and accuracy of the robot's perception play a huge role in defining the characteristics of other variations of the SLAM implementation. Therefore, a thorough understanding of advantages and disadvantages of each common perception device is an imperative concept not just for the robot's components but also the structure of the SLAM's backend algorithm.

Firstly, acoustic sensors are widely used across the preliminary stage of SLAM implementations to minimize the pose drift with time, with most of the sensors being SONAR, or Sound Navigation and Ranging [38]. These sensors are well operated in dark environments, as well as dusty and humid, due to their insensitivity towards illumination and opaqueness [39].

Secondly, LiDAR, or Light Detection and Ranging Sensor, is relatively similar to an ultrasonic sensor in terms of functionality [38]. However, LiDAR uses electromagnetic waves as a radiation reference instead of acoustic waves. A LiDAR renders a 3-dimensional representation of its surroundings known as the Point Cloud [40]. The strength of LiDAR is that the sensor can provide 360 degrees of perception with high precision [41].

Thirdly, depth cameras' mechanism works based on the illumination of the site with infrared light and measures the time-of-flight [42]. Comparing the range of measurement and accuracy, a depth camera performs poorer than a 3D LiDAR scanner because the depth camera can only acquire data within a limited range of field of view; moreover, environmental factors may affect the accuracy of the depth camera; for example, the depth camera's output is susceptible to certain materials of surfaces, such as reflective or transparent materials [43]. However, a depth camera is still a popular option for SLAM as it's a relatively economical device compared to its relatives, 3-D LiDAR, for instance.

Ultimately, event-based cameras present the local bitmap-level motion alterations to an event that took place, which is different from conventional framing-based cameras [38]. The new technique has gained popularity more recently in the field of SLAM as an event-based camera yields more efficient computational performance and better overall accuracy [44].

After reviewing the different sensors and addressing technological advancements available in today's world, it is crucial to understand how researchers have implemented those ideas to different variations of SLAM. This understanding helps in overcoming challenges and limitations that their predecessors had faced and set new standards for new research frontiers. Moreover, it becomes essential to effectively classify those SLAM variations under different criteria.

Li et al.[45] perfectly encapsulated how SLAM techniques can be classified:

Simultaneous Localization and Mapping (SLAM) techniques can be categorized by using different factors. Firstly, they can be divided into categories based on the type of sensors employed. They may include vision-based SLAM using cameras, LIDAR-based SLAM using LIDAR sensors, and RGB-D SLAM, which combines RGB cameras with depth sensors. Secondly, feature-based SLAM, which tracks distinguishing characteristics and direct SLAM, which executes mapping intensity or depth directly can be considered as different categories. Thirdly, the estimated approach, such as filter-based SLAM, which uses filters such as Particle Filter and graph-based SLAM, which is formulated as a graph optimization problem, provides another classification criterion. Finally, SLAM can be categorized based on time synchronization, with offline SLAM processing data in batches after collection and online SLAM estimating pose and map incrementally in real-time.

LiDAR-based SLAM is one of the most widely used variations of SLAM, leveraging LiDAR sensors to accurately localize itself while simultaneously building a map of its surroundings. This approach uses registration algorithms, such as Iterative Closest Point (ICP), to estimate relative transformations between point clouds during operation [46]. Feature-based algorithms, such as LiDAR Odometry and Mapping (LOAM), further enhance this process by representing 2D or 3D point cloud maps as grid maps [47]. LiDAR's ability to function reliably in diverse lighting conditions and environments makes it particularly suitable for SLAM applications in challenging scenarios.

An evolution of SLAM techniques is the use of graph-based SLAM, which focuses on optimizing a graph representation of the robot's trajectory and surrounding environment. In graph SLAM, nodes represent robot poses or landmarks, while edges denote constraints, such as relative transformations obtained from sensors like LiDAR or odometry [48]. The optimization of this graph structure, often performed using techniques like the Gauss-Newton method, allows for accurate loop closure and global consistency [49].

Another emerging trend in SLAM is the integration of multi-sensor data to overcome the limitations of individual sensing modalities. For example, algorithms such as FAST-LIO combine LiDAR data with inertial measurements from IMUs to improve robustness and accuracy [50]. Multi-sensor approaches ensure better adaptability to real-world conditions, enabling more precise localization and mapping.

Cartographer is a versatile open-source SLAM library developed by Google, designed to handle 2D and 3D SLAM applications. It is particularly known for its robust pose graph optimization and real-time performance, leveraging submapping techniques to efficiently manage computational resources. Cartographer uses LiDAR, IMU, and odometry data to create globally consistent maps by detecting and correcting loop closures, making it suitable for indoor and outdoor mapping tasks [51]. Its modular architecture allows seamless integration into various robotic platforms, making it a popular choice for research and industry applications.

Another noteworthy SLAM framework is the SLAM Toolbox, an advanced open-source library tailored for ROS 2. It provides tools for lifelong mapping, localization, and pose-graph optimization. SLAM Toolbox supports features like merging maps, multi-session mapping, and robust handling of large-scale environments. Its loop closure detection and optimization capabilities enhance global consistency, ensuring accurate localization in dynamic and complex settings [52]. This makes SLAM Toolbox ideal for applications that require long-term operation in evolving environments, such as warehouses and public spaces.

## 2.4 Collective Movement

In the domain of swarm robotics, collective movement coordination and dynamic role assignment are crucial for enabling robots to work together efficiently. Research on coordinated motion in swarms often emphasises the need for algorithms that allow robots to adapt their roles and behaviours in real-time. For example, the study on "Efficient Strategies for Coordinated Motion and Tracking in Swarm Robotics" is a comprehensive overview of various coordination algorithms, contrasting different techniques for multi-robot collaboration.

The first coordination algorithm mentioned is the leader-follower model. This algorithm is rather straightforward in the sense that one or more robots are designated to guide the swarm while the other robots adjust their positions. The leader can be pre-programmed or autonomously chosen depending on the path while the followers maintain a set distance and set angle. This model as mentioned before is simple to apply while also being centralised providing clear direction for the followers. Additionally, the followers do not need the full knowledge of the environment meaning that this model can be scalable. However, this swarm being centralised means that it is prone to a single point of failure and having reduced flexibility [53]. This model would only work well for a simple structured environment with predefined paths which unfortunately does not match with our objectives.

Another coordination algorithm is the potential fields algorithm. This algorithm is based on virtual forces with each robot in the swarm being treated as a particle that is influenced by virtual forces exerted by other robots, obstacles and targets. These forces can attract or repel each other. The object is for the robot to be "pulled" towards the goal while avoiding collisions. This model has a couple advantages; namely: Decentralised control as each robot moves autonomously based on the forces acting on it, and smooth movements. However a couple of challenges come with it as well. One challenge is the possibility of the robot being stuck in a local minimum where virtual forces cancel each other out. Secondly, proper fine tuning of the force parameters is required to prevent the robot from oscillating [54]. Overall this approach could be useful for environments with many obstacles where smooth and continuous navigation can be important. The third algorithm offered is the virtual force algorithm which is similar to the potential fields algorithm but with more constraints thereby being ineffectual to our project [38].

When comparing these three algorithms above, potential field algorithm and virtual force algorithm are decentralised while the leader-follower model is centralised. While

the leader-follower model offers a simple, scalable nature, it introduces a single point of failure. In contrast, the potential fields algorithm offers a more decentralised approach, which is better suited for dynamic environments but requires careful tuning to avoid local minima.

In swarm robotics, dynamic role assignment plays a crucial role in enabling robots to adapt their behaviours and tasks in real-time. One common method is insect-inspired behaviour, which mimics the role distribution seen in social insect colonies [55]. In this approach, robots assume roles based on simple, local rules, such as task demand or proximity to a target, without centralised control. This method offers high scalability and robustness, as robots can seamlessly take on different tasks as needed, making it suitable for large swarms. However, its reliance on local information can sometimes lead to suboptimal task assignments, particularly in complex environments where global awareness might be needed.

On the other hand, market-based approaches [56] use a more structured mechanism where robots bid for tasks based on their capabilities and availability. This ensures that tasks are allocated to the most suitable robots, leading to more efficient task execution. However, the bidding process requires communication between robots, which may introduce delays and increase system complexity. While market-based approaches tend to be more optimal for task allocation, they may not scale as easily as insect-inspired methods, especially in large or dynamic environments where constant communication is challenging.

Decentralised control in swarm robotics offers several key advantages, particularly in terms of scalability, robustness, and adaptability [57]. In decentralised systems, each robot operates autonomously, relying on local information and interactions with neighbouring robots, which eliminates the need for a central controller. This allows the swarm to scale more easily, as adding more robots does not increase the computational or communication burden on a single entity. Additionally, decentralised systems are more robust, as the failure of one or more robots does not compromise the entire system; each robot can continue functioning independently. This is particularly advantageous in dynamic or unpredictable environments, where flexibility and fault tolerance are critical.

In contrast, centralised control systems rely on a single controller to manage all robots, which creates a bottleneck as the number of robots increases. Centralised systems can suffer from single points of failure—if the controller fails, the entire system may halt. Moreover, communication delays and computational limits can hinder real-time performance in larger systems. While centralised control offers more efficient coordination in smaller, simpler environments, decentralised control is better suited for real-world applications where scalability and resilience are essential for handling complex and dynamic tasks.

Despite significant advancements in swarm robotics and coordination algorithms, there remain notable gaps in applying these methods to practical, real-world environments such as cleaning tasks. Most research on algorithms like potential fields, leader-

follower models, and market-based role assignment has focused on simulations or controlled environments, which often lack the complexity and unpredictability found in real-world scenarios. For instance, limited work has been done on integrating these algorithms with sensor-rich, dynamic settings where robots must navigate cluttered spaces, identify and manipulate diverse objects, and coordinate in real-time without centralised control. Additionally, the scalability of these systems is often not tested in practical, large-scale environments, such as a commercial building cleaning system, where communication constraints, battery life, and real-time decision-making are crucial factors.

Our project aims to address these gaps by developing a swarm of cleaning robots that leverages C-SLAM for real-time mapping and localization, along with a hybrid role assignment approach that adapts based on task proximity and robot capabilities. By testing in realistic environments with obstacles, diverse object types, and multiple robots working simultaneously, our project will not only explore the robustness of these algorithms but also refine them for practical deployment in domestic and commercial cleaning. This will bridge the gap between theoretical research and real-world application, offering a scalable and adaptive solution for multi-robot systems.

# Chapter 3

## Hardware

This chapter presents the practical implementation of the Jeff3 robot, highlighting its transition from a purely simulated environment in Webots to a fully functional physical system. Last semester, the focus was on simulation to establish the foundation of decentralized swarm communication, with no hardware demonstration. This semester, the work has progressed significantly, with all developments extended to physical hardware. Jeff3, shown in Figures 3.4 and 3.3, was originally designed using CAD software and has now been realised as a working prototype.



Figure 3.1: Final Presentation with 2 robot present

This chapter details the practical implementation of the robot Jeff3, transitioning from purely simulated in Webots setup into physical, functional robotic system. This chapter will be covering the work done, and the knowledge accumulated in the hardware section. Last semester no demonstration was done in hardware, all was done in simulation to focus on the communication which is the foundation of a decentralized robotics swarm system. This semester all the work done have been extended and implemented into hardware. Robot Jeff3, depicted in Figures 3.1 and 3.2, was initially designed in CAD software during the previous semester and transitioned into a physical prototype this semester.

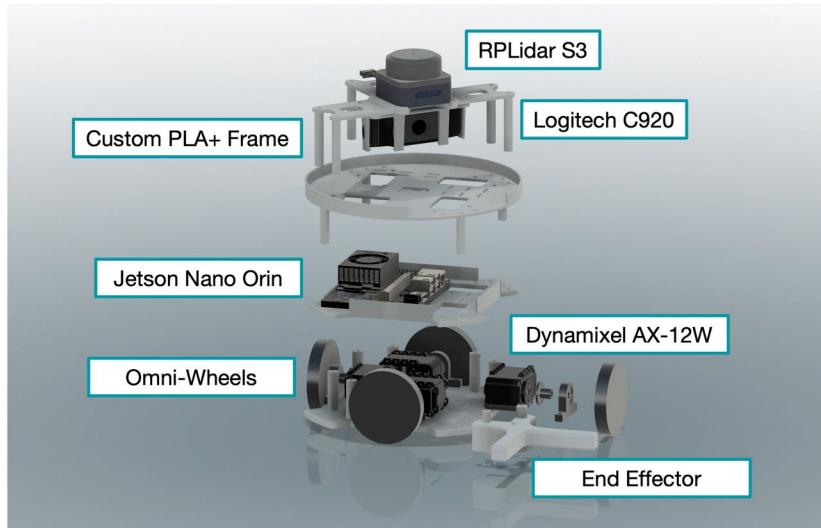


Figure 3.2: CAD Model of Robot Jeff3 annotated

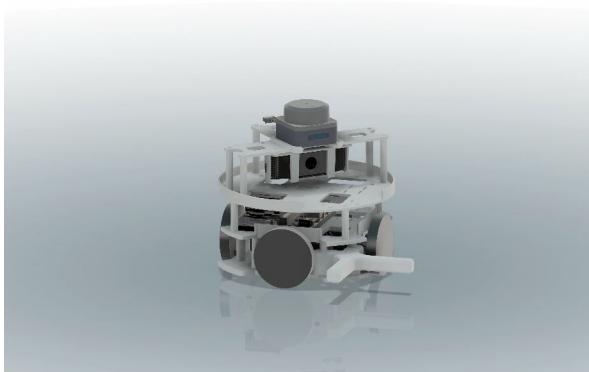


Figure 3.3: CAD Model of the designed robot named Jeff3

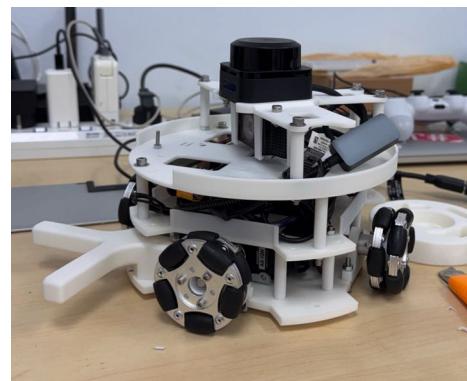


Figure 3.4: Final Form of the Jeff3 Robot

### 3.1 Hardware Layout

The test arena was specifically designed to minimize external disturbances, crucial for accurate robot localization and movement. The test arena can be seen in 3.5 The requirements for the hardware are the following:

1. **Location:** Lab where a sterile arena can be implemented with minimal interruptions, Chulaprat 14
2. **Size:** 3-4 square meters of space, 2.1m x 1.4m
3. **Floor:** Non-Reflective Floor to optimize for the computer vision system & reducing false positive, White Paper floor
4. **Barrier:** Barrier against a runaway robot, abandoned Metal Boxes

In addition to the layout, a C-Stand supporting a Logitech C920 webcam for robot position validation and camera odometry. More information are provided in the Localization section.

## 3.2 Robot Frame

In last semester midpoint point we have planned out the hardware contruction in CAD before developing in hardware, In Figure3.3, is the was model, the footprint of the robot is small at 20 cm in diameter and 22cm in height.



Figure 3.5: Test Arena layout for the swarm project with a camera setup

PLA+ frame is printed with a Bambu Lab P1S. PLA+ is chosen due to the ease of printing, eco-friendly, stiffness. Other material such as ABS and ASA or PETG, however, these release toxic fumes, flex more than PLA+. Initially, a 5 mm thick PLA base was chosen, providing adequate rigidity; however, future tests will evaluate if a 5 mm acrylic base could enhance structural stiffness. As according to a paper, 3D printed PLA has a range of Young's modulus of 2.1 to 4.4 GPa, while acrylic has a Young's modulus of 3.6 GPa. Therefore, hard to conclude with this better without proper testing. 3.4 is the final forum of the robot. The rationale behind using 3D printing as an alternative to 3D printing was due to the speed at which things can be prototyped. With modern printing like a Bambulab the accuracy are very good with tolerance of 0.2 mm with good modeling design in mind with such as thermal expansion after printing. Resulting in ability to press fit the bearing all eight Bearing 626ZZ to the 3D printed bearing pillow holder. The image on the left of the 3.6(left), are 6 mm metal shaft to replace the PLA 3D printed ones, these shalft will be supported by the bearings, see 3.8.

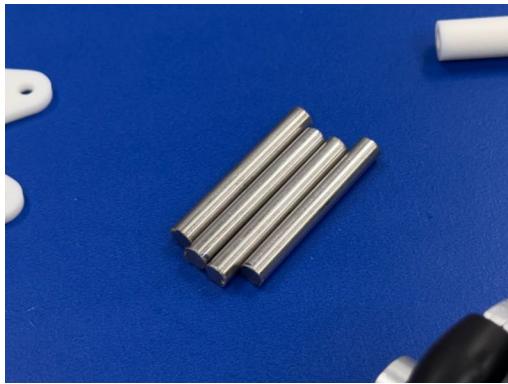


Figure 3.6: Sliced Metal Shafts

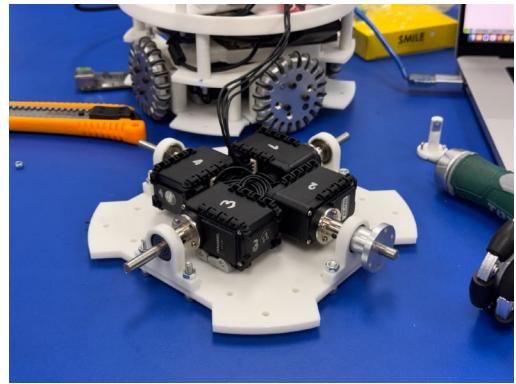


Figure 3.7: Base Level Assembly of the motors, motor flange, bearing pillow block, omni-wheel flange

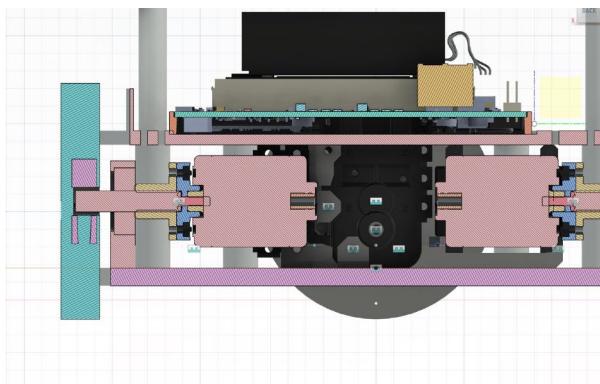


Figure 3.8: Section view of the wheel, shaft, bearing and the motor

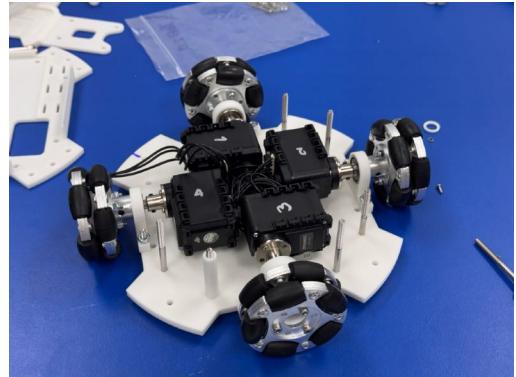


Figure 3.9: Complete base level assembly with Omnidirectional-wheels attached



Figure 3.10: Printed mount for the Lidar and C920 webcams

The robot's sensing suite comprises two primary sensors: the RPLidar S3 for navigation and obstacle detection, and the Logitech C920 webcam for visual odometry and

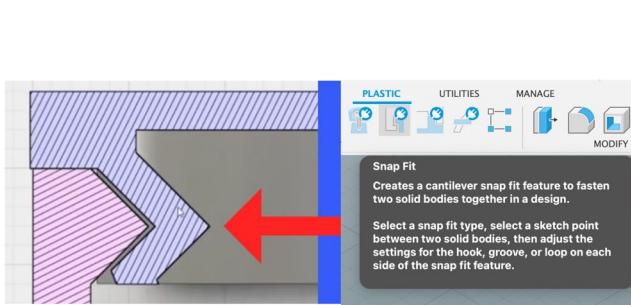


Figure 3.11: Screenshot of Fusion360 Snap Fit Feature for Plastics

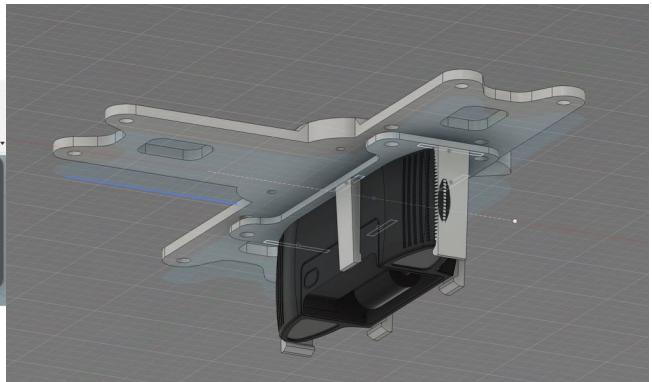


Figure 3.12: Final CAD design of the mount for both C920 Webcam and

validation; both of which require a custom 3D printed mount. The C920 had no standard mounting fitting the size of the robot, therefore require a custom mount. Measurements were taken using a vernier caliper as there's no publically available CAD model, unlike the RPLiDAR S3. With no visible mounting on the C920 the monitor stand was taken out and a 3D model of the enclosure is designed using the Snap Fit feature found in the Fusion 360(Learnt in Robotics Project IV), see 3.11 and 3.12. See the final result in Figure 3.10. The Following are the Bill of Material(BOM) cost of one robot:

1. 4x Dynamixel Motors AX-12W - 2,100THB each
2. 4x Omnidirectional wheels - 980THB each
3. 4x Bearing 626ZZ 06x19x06mm - 30THB each
4. 1x RPLidar S3 Lidar - 23,850THB each
5. 1x Logitech C920 RGB Camera - 3,460THB each
6. 1x Jetson Orin Nano 14,600THB each
7. 1x USB - TTL board - 400THB each

Total 48,454THB per robot, important to iterate this a proof of concept of a decentralized system, not for commercialization.

To showcase collective transport we are sliding the cylinder synchronously not needing a complex system to grip the robot. The End effect used is a simple fixed the frame, it's a right angle gripped for the testing purposes it's sufficient in gripping a cylinder when 2 or more robot are present. the CAD and printed model can be seen in 3.13 and 3.14

### 3.3 Robot Drive Train and kinematic

The motors used are the Dynamixel, In Figure 3.16, are the Inverse Kinematic our robot. The requirement for omnidirectional wheels are due to the collaborative step in the demo, where the 2 robot must move the same time such as diagonally without non-holonomic constraint

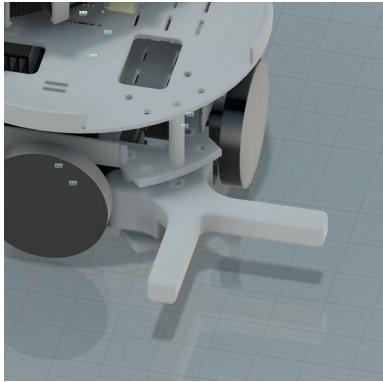


Figure 3.13: CAD Model of a gripper attached to the robot

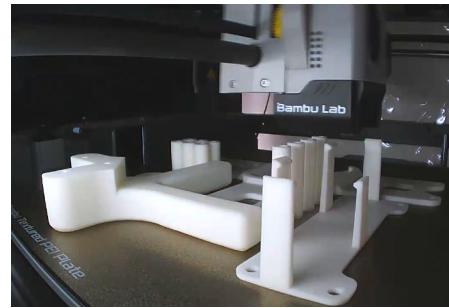


Figure 3.14: A 3D Printed gripper using Bambulab P1S

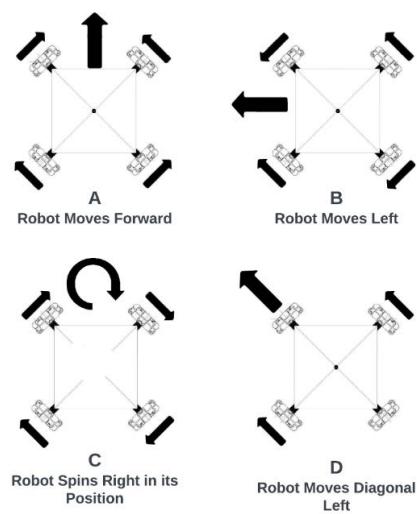


Figure 3.15: Direction of the robot movement with respect to each wheel movement

$$\dot{\mathbf{q}} = J^{-1} \cdot \dot{\mathbf{P}}$$

In this expression,  $\dot{\mathbf{q}} \in R^4$  is the vector of angular velocities of the four drive wheels, while  $\dot{\mathbf{P}} \in R^3$  is the platform velocity vector consisting of the forward velocity  $V_x$ , lateral velocity  $V_y$ , and yaw rate  $\dot{\theta}_z$ . The inverse Jacobian matrix  $J^{-1}$  relates the platform motion to the required wheel speeds, based on the robot's kinematic configuration.

For the X-drive configuration, the Jacobian takes the following matrix form:

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\sin(\theta + \frac{\pi}{4}) & \cos(\theta + \frac{\pi}{4}) & R \\ -\sin(\theta + \frac{3\pi}{4}) & \cos(\theta + \frac{3\pi}{4}) & R \\ -\sin(\theta + \frac{5\pi}{4}) & \cos(\theta + \frac{5\pi}{4}) & R \\ -\sin(\theta + \frac{7\pi}{4}) & \cos(\theta + \frac{7\pi}{4}) & R \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \dot{\theta}_z \end{bmatrix}$$

Figure 3.16: Wheel velocity calculation using the inverse Jacobian matrix. The vector on the left-hand side  $[\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4]^T$  represents the angular velocities of the four wheels. The Jacobian matrix reflects the X-drive wheel configuration. In this context,  $\theta$  is the robot's global orientation,  $R$  is the radial distance from the robot's centre to each wheel, and  $r$  is the wheel radius. The input vector on the right-hand side comprises  $V_x$  and  $V_y$ , the body-frame linear velocities, and  $\dot{\theta}_z$ , the angular velocity about the vertical axis.

Each resulting wheel angular velocity  $\dot{q}_i$  is then used as a target input to a local PID controller, which ensures the wheel reaches and maintains this desired velocity. The control law is based on the difference between the target and measured angular velocities of the wheel:

$$e_i(t) = \dot{q}_i^{\text{target}}(t) - \dot{q}_i^{\text{measured}}(t)$$

The control input  $u_i(t)$ , which is applied to the motor driver, is then computed using the standard PID formulation:

$$u_i(t) = K_p e_i(t) + K_i \int_0^t e_i(\tau) d\tau + K_d \frac{d}{dt} e_i(t)$$

Here,  $e_i(t)$  represents the instantaneous control error at time  $t$ ,  $K_p$  is the proportional gain that reacts to present error,  $K_i$  is the integral gain that accumulates past error to eliminate steady-state offset, and  $K_d$  is the derivative gain that anticipates future error trends. The PID controller thus dynamically adjusts the motor actuation to match the target wheel velocity, enabling precise and stable tracking of the robot's motion commands.

## 3.4 Base Platform Modification and Microcontroller Integration

The initial version of the robot was built using Dynamixel AX-12W motors. However, we encountered major issues in odometry due to the low quality and unreliability of the built-in encoders. These limitations hindered accurate velocity and position estimation.

To overcome these issues, we redesigned the robot base. The new design incorporates DC motors equipped with optical encoders mounted on the rear shaft. This design offers improved resolution and provides reliable velocity feedback, which is crucial for accurate closed-loop control.

Figure 3.17 shows the configuration, where four DC motors operate in coordination. Each motor is driven by an L298N H-bridge motor driver. The control signals for each driver are generated by an ESP32 microcontroller.

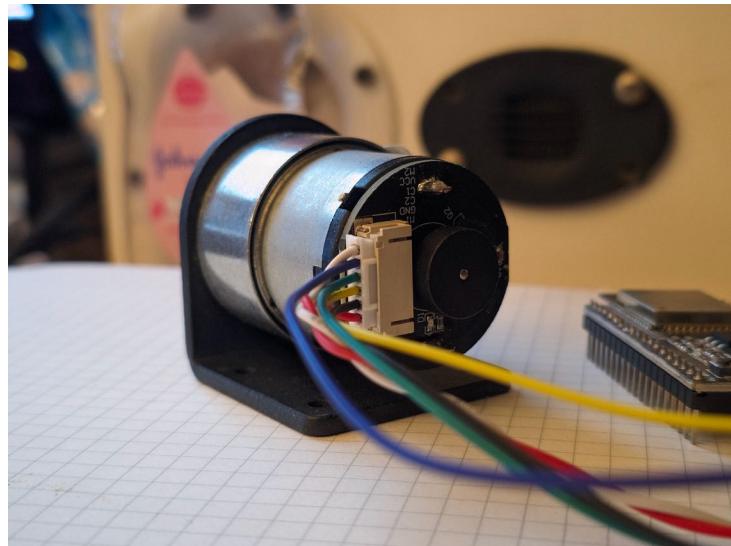


Figure 3.17: DC motor with integrated encoder at the rear shaft, enabling closed-loop control.

The ESP32 is programmed using the ESP-IDF framework, which is based on FreeRTOS. Unlike the Arduino framework, ESP-IDF provides direct access to low-level resources and supports real-time scheduling. This choice removes the overhead introduced by Arduino's abstraction layers and enables finer control over timing and concurrency. As a result, we achieve more accurate and responsive motor control. The new platform design and software stack lay the foundation for the following section, which describes the detailed architecture of the motor control system.

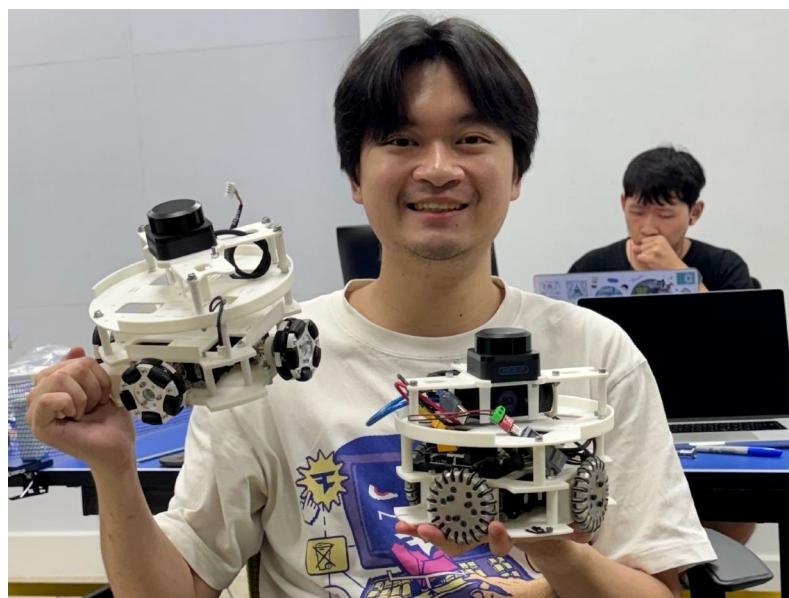


Figure 3.18: 2 built robots "Jeff3.0"

# Chapter 4

## Object Detection Using Computer Vision

Building upon our previous progress report, this section describes the use of the S3 RPLiDAR in conjunction with a webcam for detecting object width, estimating relative distance, and determining angular position of objects. The initial objective was to develop a model capable of detecting and analyzing three cylindrical objects—tested one at a time—since only one cylinder would be present in the arena at any given moment alongside three mobile robots. However, this setup was later refined: the number of cylinders used in the real-world environment was reduced to two—a blue cylinder (16 cm diameter) and a yellow cylinder (20 cm diameter). A third cylinder with a 12 cm diameter was reserved for testing the model’s flexibility and generalization.

To summarize the detection pipeline: the robot moves randomly until a target cylinder is detected. Detection is considered valid when the center of the bounding box falls between pixel positions 600 and 680, assuming 0 is the leftmost pixel. Once this criterion is met, a bounding box is drawn, the system state is set to `FoundObject`, and the robot halts for three seconds. During this pause, the system calculates the relative distance (from the robot’s center to the object surface at the bounding box center), the relative angle, and the estimated diameter. The workflow is illustrated in Figure 4.1.

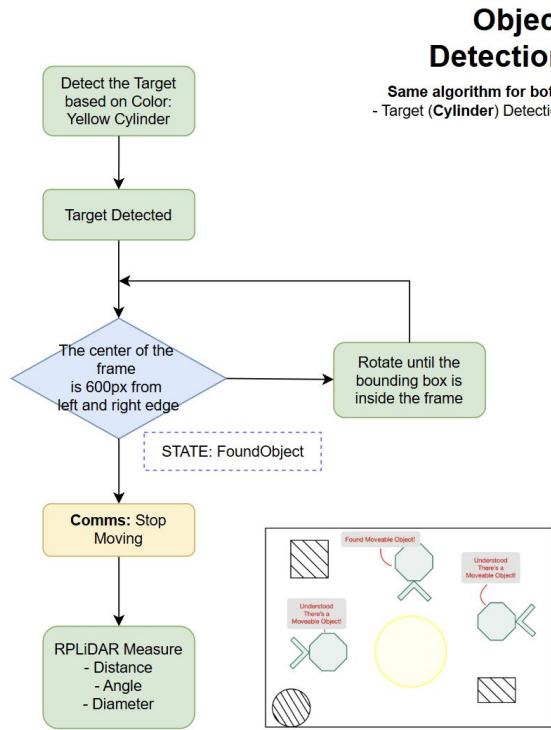


Figure 4.1: Overview of the object detection and measurement workflow

Object detection is performed using color segmentation based on the YCbCr color space. Only two colors—blue and yellow—were tested, each defined by specific value ranges within the YCbCr space. The range for blue is [0, 100, 140] to [255, 255, 255], and yellow is [100, 140, 50] to [255, 255, 100].

To enable accurate measurement and computation, multi-sensor fusion is implemented. A key factor in this process is the horizontal field of view (FoV) of the camera, determined through calibration using OpenCV. The calibration process uses images of a checkerboard pattern to estimate the camera's intrinsic parameters. The horizontal FoV was calculated to be 72 degrees.

Mapping the object's angular position from the camera frame to the LiDAR frame is achieved using the equation:

$$\text{relative\_angle} = \left( \frac{\text{object\_center\_x} - \text{frame\_center\_x}}{\text{frame\_width}} \right) \times \text{horizontal\_fov} \quad (4.1)$$

Once the relative angle is computed, the LiDAR provides the corresponding angle and distance values, using data retrieved via the `rclpy` library in ROS 2.

With the relative distance and angle known, the cylinder's diameter is estimated using an image projection method, shown in Figure 4.2.

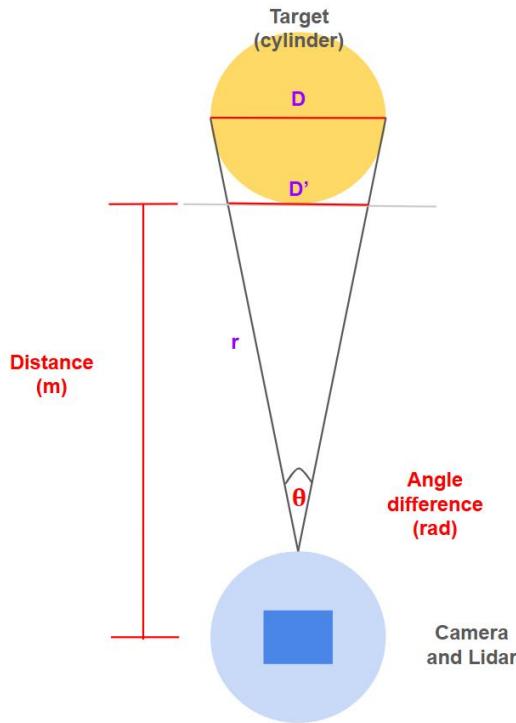


Figure 4.2: Image projection visualization

The estimation begins by calculating the distance  $r$  from the LiDAR to the object's side using:

$$r = \frac{\text{distance}}{\cos\left(\frac{\theta}{2}\right)} \quad (4.2)$$

Here,  $\theta$  is the angle subtended by the object in the camera frame. Then, the projected diameter  $D'$  is computed using the law of cosines:

$$D' = \sqrt{2r^2 - 2r^2 \cos(\theta)} \quad (4.3)$$

The projected radius is:

$$w' = \frac{D'}{2} \quad (4.4)$$

Using geometric similarity, the actual radius  $w$  is calculated as:

$$w = \frac{\text{distance} \cdot \tan\left(\frac{\theta}{2}\right)}{1 - \tan\left(\frac{\theta}{2}\right)} \quad (4.5)$$

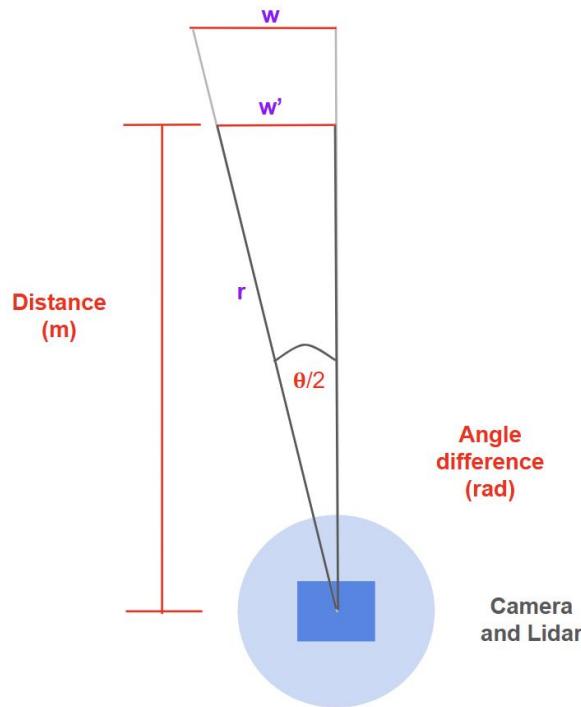


Figure 4.3: Calculation of radius from projected image

Although the diameter  $D$  is calculated, camera distortion introduces an error of approximately  $\pm 4$  cm for distances ranging from 0.4 to 3.0 m. This justifies the constraint on the detection zone to maintain real-world angular accuracy. The calculated diameters are compared against actual measurements in Figure 4.4. The model achieved a Mean Absolute Error (MAE) of 0.01289, Mean Squared Error (MSE) of  $2.902 \times 10^{-4}$ , and R-squared of 0.0431.

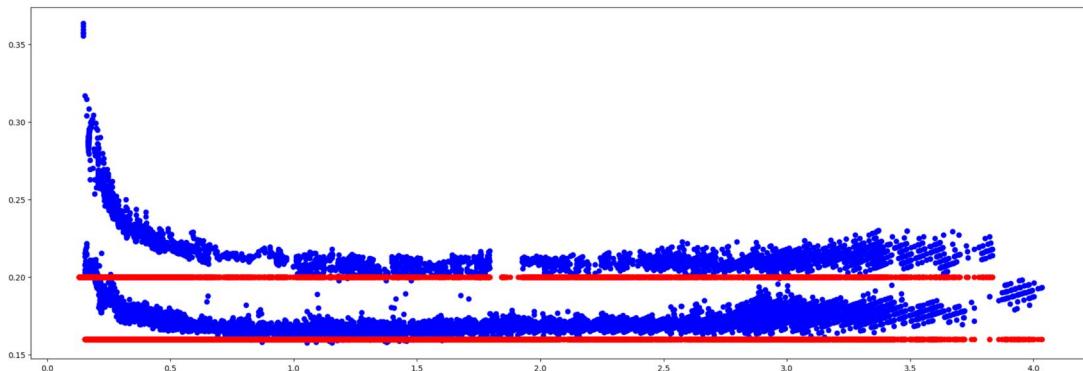


Figure 4.4: Comparison of calculated (blue) vs actual (red) diameters

To minimize diameter estimation error, a variation factor is introduced:

$$w' = \frac{\text{actual\_width}}{D} \quad (4.6)$$

The variation factor is computed for each actual-estimated diameter pair. These values are used to train a predictive model using `scikit-learn`, with distance as the independent variable. 80% of the dataset is used for training and 20% for testing. A sigmoid function is fitted to model the relationship, as shown in Figure 4.5.

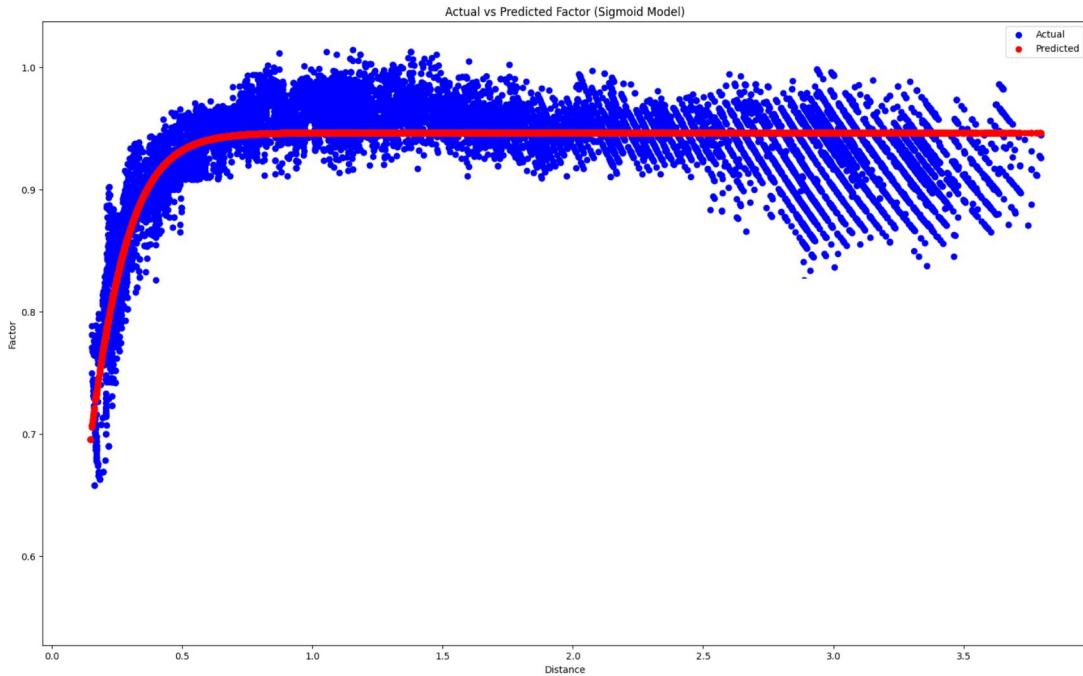


Figure 4.5: Fitted sigmoid function for variation factor

The final form of the fitted sigmoid function is:

$$\text{factor} = \frac{0.9461218686}{1 + e^{-9.0309891171(x - 0.0335094048)}} \quad (4.7)$$

Model performance on the test set is evaluated using the following metrics:

Table 4.1: Model Accuracy Metrics

Metric	Value
Mean Squared Error (MSE)	$2.802 \times 10^{-5}$
Mean Absolute Error (MAE)	0.004
R-squared ( $R^2$ )	0.916

The final cylinder diameter is obtained by multiplying the variation factor by the value of  $D$ . The prediction accuracy is visualized in Figure 4.6.

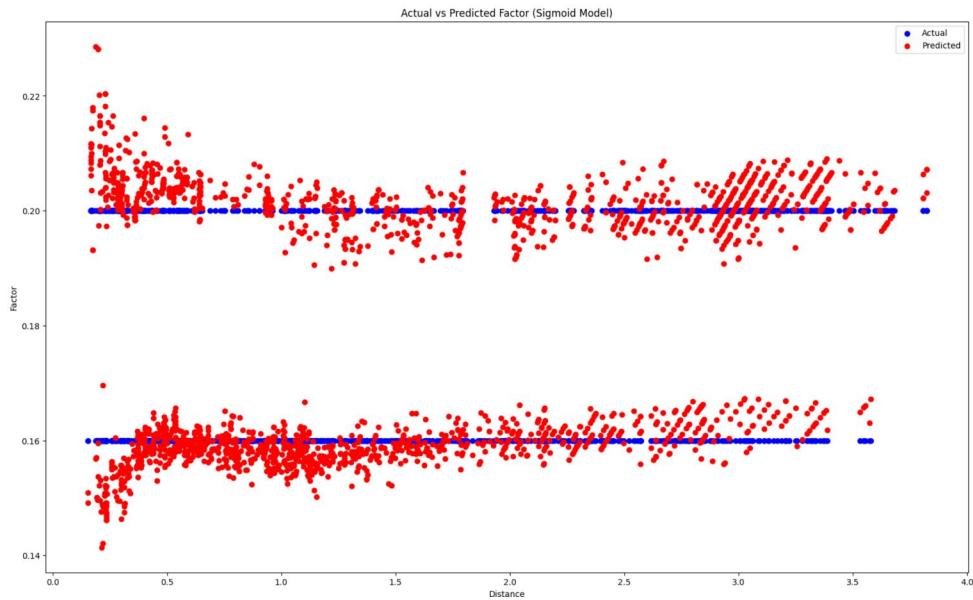


Figure 4.6: Predicted vs actual cylinder diameter

Model robustness was evaluated by placing cylinders at distances of 0.5 m, 1 m, 2 m, and 3 m under nine different lighting conditions, shown in Figure 4.7. Lighting conditions 1–8 involved radial illumination at 0.4 m and a 30-degree elevation angle. Condition 9 featured overhead lighting from a height of 0.3 m. All tests were conducted in a darkened lab environment.

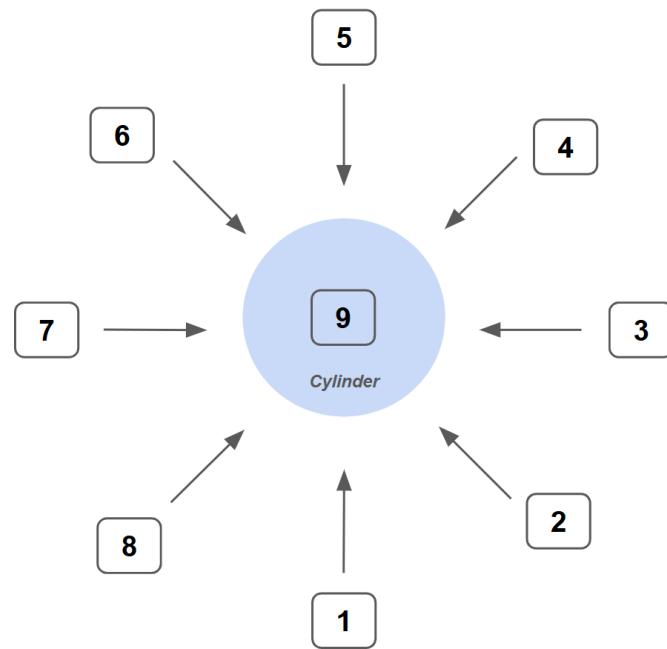


Figure 4.7: Nine lighting conditions used for model testing

Measurements for each cylinder under all lighting conditions are presented in Appendix A. Tables A.1 and A.2 show predicted diameters, relative distances, and angles

for the 20 cm and 16 cm cylinders, respectively. Table A.3 evaluates the model's generalization on the 12 cm cylinder. Summary errors are shown in Table 4.

<b>Table</b>	<b>Mean Absolute Width Error (m)</b>	<b>Mean Absolute Angle Error (°)</b>
Table 1	0.0022	0.2389
Table 2	0.0017	0.2219
Table 3	0.0061	1.0519

Table 4.2: Mean absolute width and angle errors

After applying a variation factor to optimize the accuracy of width measurements, the model's performance improved significantly. The prediction error was reduced from 4 centimeters to approximately 1 centimeter for cylinders with diameters of 20 cm and 16 cm, which are the objects intended for use in the actual test environment. In contrast, the 12 cm diameter cylinder exhibited a higher prediction error of up to 2 centimeters.

Two major challenges were identified: camera distortion and hardware-related issues. Firstly, camera distortion significantly hindered the accurate determination of the target's diameter. Lens distortion caused the image to expand both horizontally and vertically, resulting in inaccurate distance and width measurements, with a predicted width error of 4 centimeters. To mitigate this issue, limiting the tolerance zone to a specific range in the center of the frame effectively eliminated distortion on both sides.

Another challenge was hardware-related. The original CSI camera could not be utilized due to a firmware update on the Jetson Nano, which lacked support for this type of camera. Additionally, the camera occasionally failed to auto-focus, causing blurred images and inaccurate measurements, or no results at all. This issue occurred when objects were very close to the lens, specifically within 30 centimeters. To address this, the status will not be set as "FoundObject" if the distance is less than 30 centimeters.

To further enhance the model's flexibility and accuracy, it is recommended to incorporate data from the 12 cm cylinder into the determination of the variation factor function. Ultimately, to generalize the model for different cylinder sizes or even other geometric shapes, the variation factor should be defined as a function of intrinsic and extrinsic camera parameters, which can be accurately obtained through OpenCV camera calibration.

# Chapter 5

## Communication in the Swarm

Swarm communication has been successfully implemented using a **peer-to-peer socket communication model**, where each robot can directly exchange data with others without relying on a centralized server. This architecture was chosen for its low-latency characteristics, which are crucial for real-time robotic coordination. While raw socket communication is common, the peer-to-peer topology enables direct robot-to-robot interaction, reducing delays caused by intermediaries. To improve reliability, our implementation includes:

1. Connection checks with configurable retry intervals and timeouts triggered only when broadcasting (Algorithm 1)

---

### Algorithm 1 Connection Check and Reconnection

---

```
1: Input: target_robot_ip, message
2: if message is ready to be broadcast then
3:   if not connected to target_robot then
4:     for attempt = 1 to MAX_RETRIES do
5:       Try to connect to target_robot_ip
6:       if connected then
7:         break
8:       else
9:         wait for RETRY_INTERVAL
10:      end if
11:    end for
12:  end if
13:  if connected then
14:    Broadcast message
15:  end if
16: end if
```

---

2. Retry logic inspired by a three-way handshake (ACK/SYNACK) (Algorithm 2)

**Algorithm 2** Three-Way Handshake Inspired Message Broadcast and Retry Logic

---

```

1: Input: message, target_robot
2: Send message to target_robot
3: Start ACK_timer
4: while ACK not received and ACK_timer not expired do
5:   if ACK or SYNACK received then
6:     Confirm delivery
7:     break
8:   end if
9: end while
10: if ACK not received then
11:   Retry or log transmission failure
12: end if
```

---

3. Structured message formats with headers for message categorization and handling (Algorithm 3)

**Algorithm 3** Message Parsing and Handling

---

```

1: Input: received_message
2: Extract header from received_message
3: if header.type is "COORDINATES" then
4:   Update robot coordinates
5: else if header.type is "OBJECT_DETECTED" then
6:   Trigger task assignment handler
7: else if header.type is "PATH" then
8:   Extract path and orientation from received_message
9: else
10:  Ignore or log unrecognized message
11: end if
```

---

Within the swarm, this communication setup allows continuous exchange of positional data for effective self-collision avoidance and enables robots to relay relevant information to the designated taskmaster for path planning. The overall communication flow is illustrated in Figure 5.1.

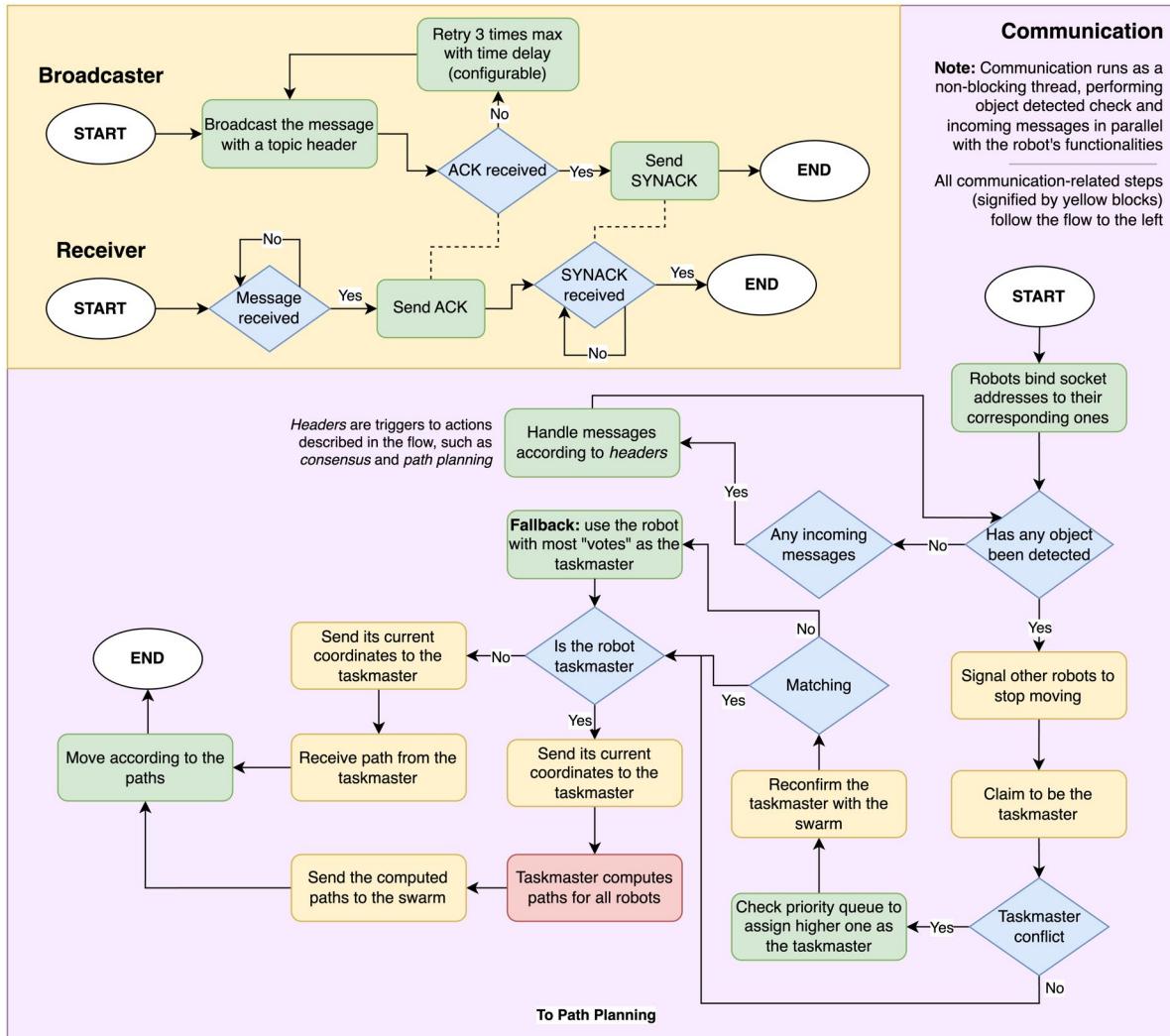


Figure 5.1: Communication in the Swarm Detailed Flow

There are two main implementations using communication mechanism in the swarm. The implementations are the "**constant coordinate stream**", and the "**swarm detection task flow**". For simplicity, all future descriptions of the communication mechanism will refer to the robots as "*Robot A*", "*Robot B*" and "*Robot C*".

The **constant coordinate stream** is a stream of coordinates in a fixed, configurable interval between the robots A, B, and C. They share their current X and Y positions with one another, updated by data obtained from the odometry. It is essential for the robots in the swarm to know the positions of one another so the movement controller will allow them roam in a collision-free manner. Figure 5.2 is a simple display of the constant coordinate stream in action, providing an overview to the data the robots will obtain from this coordinate stream. The output is obtained from Robot B, where it is able to continuously track the coordinates of Robot A, Robot C, as well as itself, even with the others' coordinates being updated from their respective movements.

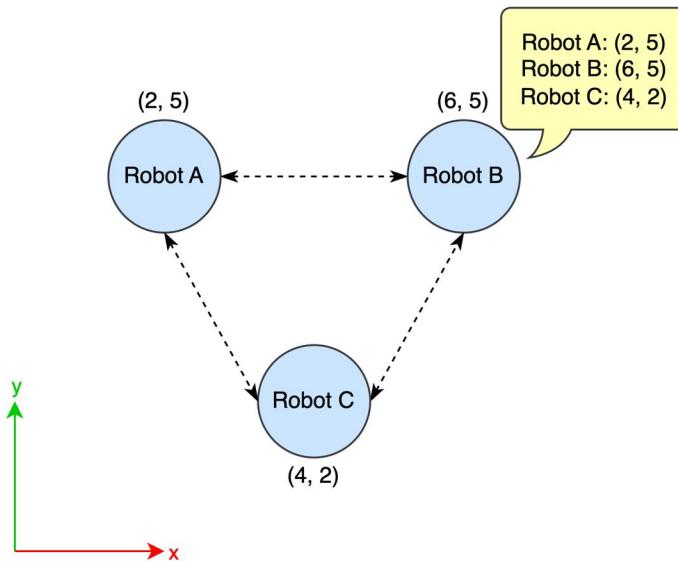


Figure 5.2: Constant Coordinate Stream

The other essential communication implementation is the **swarm detection task flow**. When an object gets detected by the robot's object detection, the detecting robot will send a signal to the other two robots to stop moving. Figure 5.3 illustrates an instance where Robot A detects an object and claims to be the taskmaster. Robot B and Robot C will receive a signal to stop ongoing movement (Figure 5.4) and await a collisionless path to move towards the object by the swarm's path planning algorithm, detailed in a later chapter.

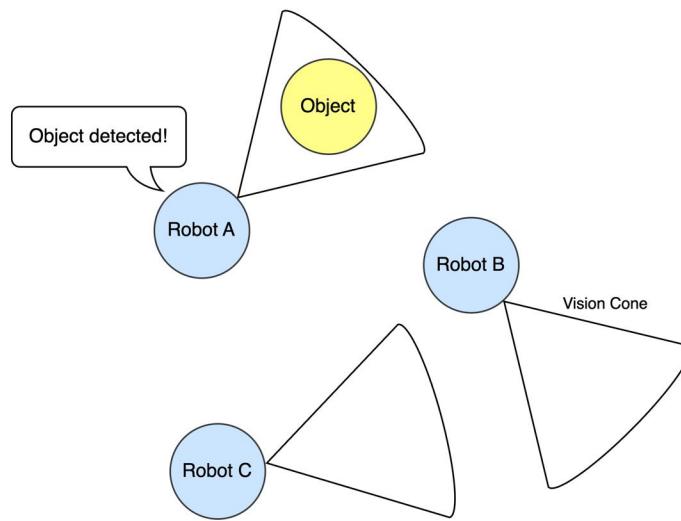


Figure 5.3: Swarm Detection Task Flow – Stage 0

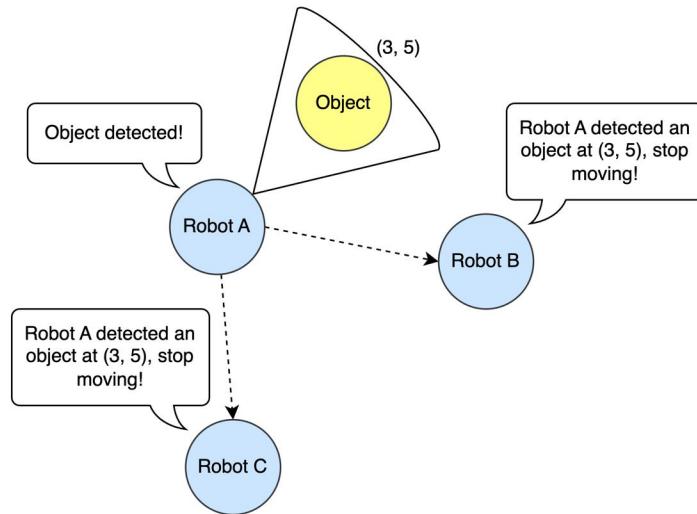


Figure 5.4: Swarm Detection Task Flow – Stage 1

The taskmaster, in this case, Robot A, will request current coordinates from Robot B and Robot C (Figure 5.5), then trigger the path planning algorithm for a set of collisionless paths for each robot. Once the set of paths are computed by the taskmaster, it will subsequently broadcast them towards others in the swarm. In this case, Robot A performs the computation and broadcasts to Robot B and Robot C, allowing all of them to have their respective determined paths to move towards the detected object (Figure 5.6)

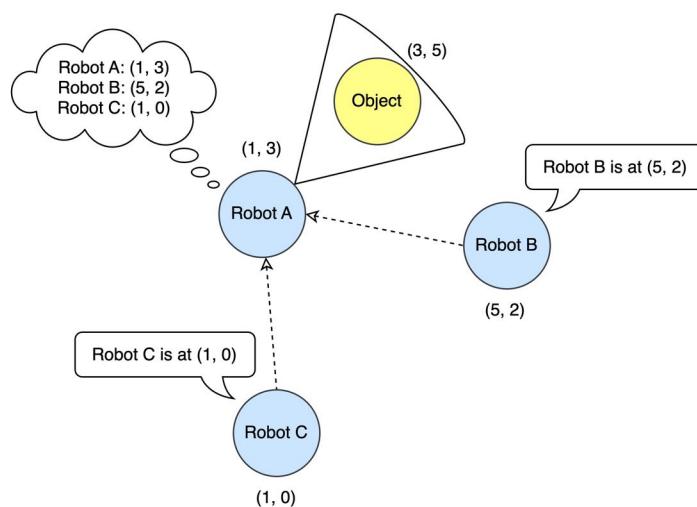


Figure 5.5: Swarm Detection Task Flow – Stage 2

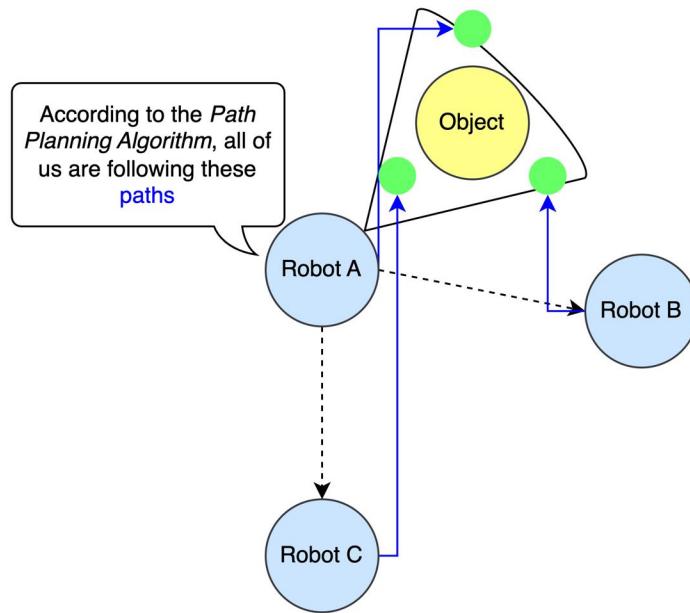


Figure 5.6: Swarm Detection Task Flow – Stage 3

Another thing to note is that any swarm member can become the taskmaster and perform the computations according to the *swarm detection task flow*. In the described scenario, Robot A is the member that detect the object and became the taskmaster. If Robot B or Robot C were to detect the object first, they can also claim to become the taskmaster and computational lead of the swarm detection task flow. This is designed to enhance the versatility in the communication mechanism of the swarm.

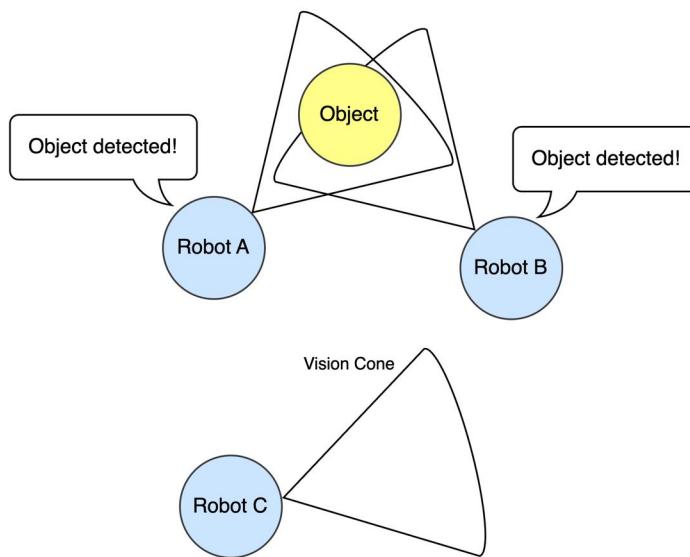


Figure 5.7: Swarm Detection Task Flow – Conflict

Since there is a claiming mechanism to become the taskmaster in charge of path computations, it is imperative to design communication in such a way that any *conflicting claims* in the exact moment are handled appropriately. An instance of *conflicting claims* is when multiple robots detect the object at the same time, hence claiming to become the taskmaster, similarly to the situation portrayed in 5.7. The handler of this situation is the designed **consensus algorithm** for the swarm. Figure 5.8 outlines the underlying mechanism of the consensus algorithm.

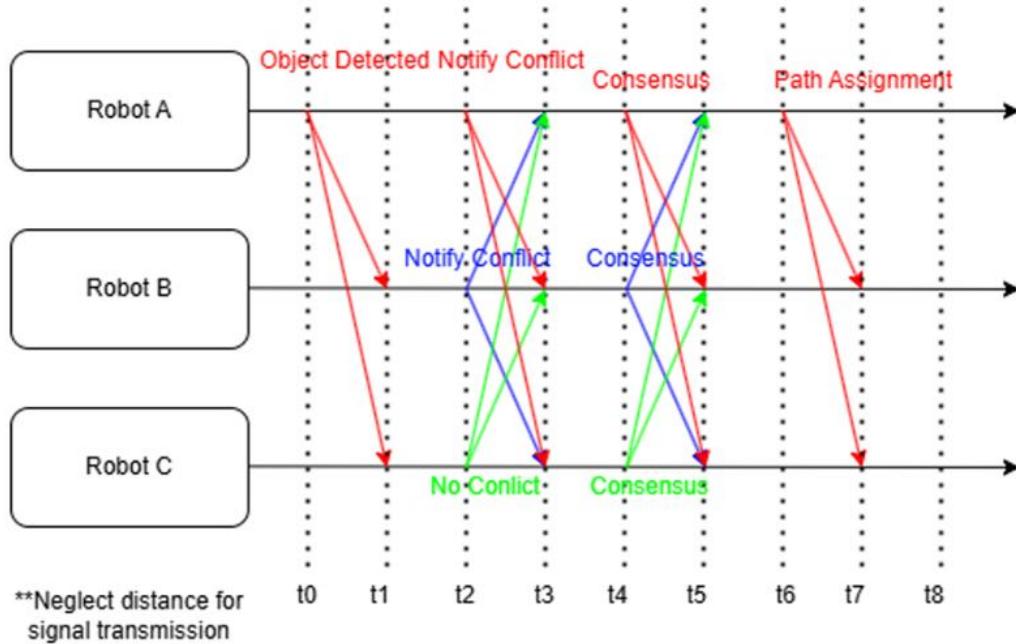


Figure 5.8: Consensus Algorithm

The consensus algorithm functions by referring to a *priority queue* once conflicting claims are observed. This priority queue determines the taskmaster by appointing the claimant with the higher priority the taskmaster role. Subsequently, the claimant is repositioned to the end of the priority queue for future claims to partially decentralize the operation. An account of this consensus mechanism is when Robot A and Robot C detect an object at the same time, with Robot C being higher on the priority queue. Robot C will get the taskmaster assignment for that particular detection, then rearranged to the end of the priority queue. If Robot A and Robot C detect an object at the same instance again, Robot A will receive the role, and also gets rearranged to the end of the queue. After the taskmaster assignment resolution is complete, the *swarm detection task flow* will occur as designed.

The swarm communication approaches, particularly the *constant coordinate stream*, are primarily required to help robots efficiently utilize resources and avoid collisions, especially during random movement when no objects have been detected. The *swarm detection task flow* is designed to efficiently travel to designated coordinates in order to prepare for collective movement of the detected object by gripping. The consensus algorithm is also designed to allow seamless coordination and eliminate potential conflicts in the swarm communication.

# Chapter 6

## Coordination and Formation

This chapter presents the implementation of a swarm path planning system designed to form a coordinated structure around a target object. The formation algorithm computes collision-free trajectories based on the positional data received from each robot, assigning optimized paths, typically the shortest feasible route, to individual swarm members. These paths are then converted into discrete waypoints, a format well-suited for execution by the swarm's motion controllers, ensuring accurate and efficient formation around the object. The flow for formation planning is illustrated in Figure 6.1.

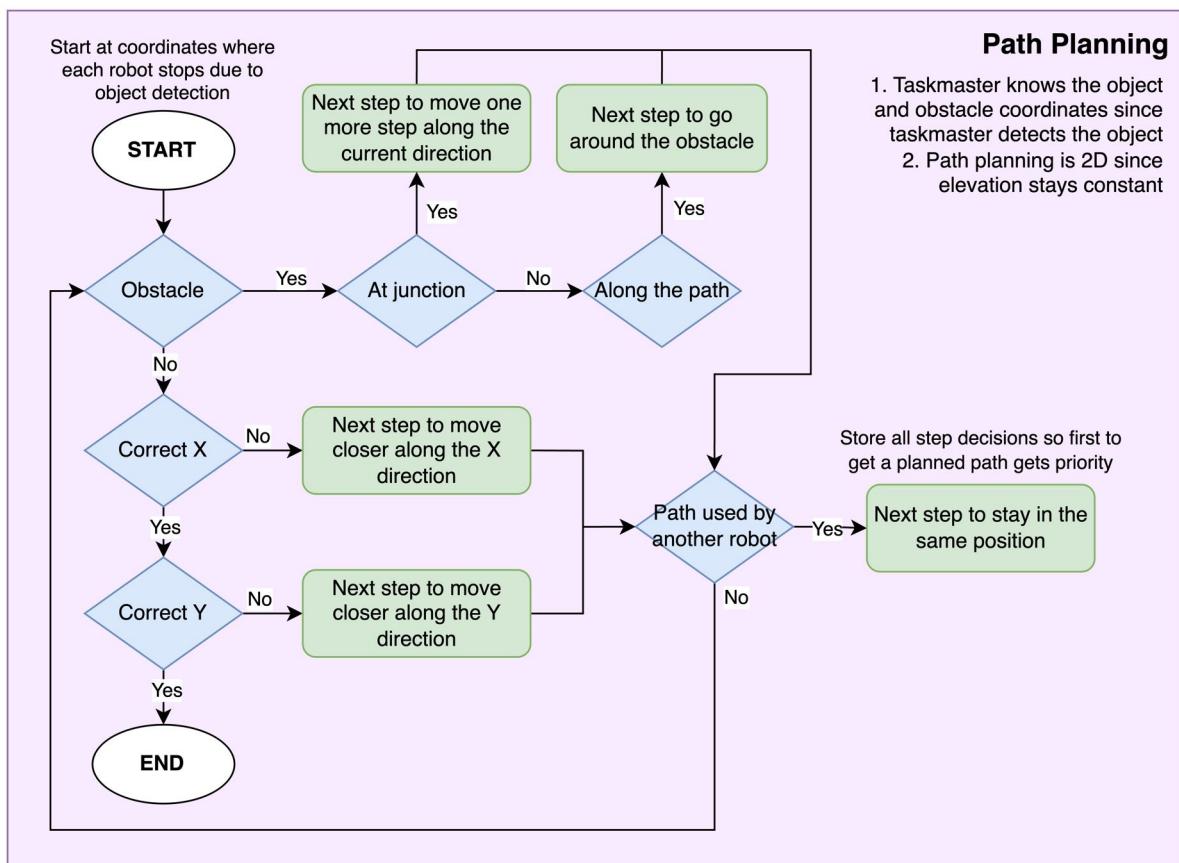


Figure 6.1: Path Planning Flow

In the process of building a swarm formation, computations for the desired coordinates where the swarm units should be moving to (later referenced as "**target coordinates**"), and the desired path the members can take in order to move towards the goal are essential. The key parameters for this calculation include: the **radius** where the swarm should place itself around the object, swarm fleet **member count**, **current coordinates** for the swarm, and target **object coordinates**.

The member count is a vital initial point since it will be used to compute the angle  $\theta$  between each set of target coordinates using the following formula:

$$\theta = (2\pi/n) * i$$

**where:**

$\theta$  = angle between each set of target coordinates (in radians)

$n$  = swarm system member count

$i$  = member identifier (e.g. 0 to 4 for a swarm fleet of 5 members)

Currently, the formation is determined by utilizing a given radius for the swarm robotic members to attempt to surround. In accordance with the member count providing the angle, the individual target coordinates for each robot can be calculated using an adaptation from the Pythagorean Theorem.

$$x' = x + r\cos\theta$$

$$y' = y + r\sin\theta$$

**where:**

$(x, y)$  = robot's present coordinates

$(x', y')$  = target  $(x, y)$  coordinates

$r$  = radius

The target coordinates at this stage would often result in floats; therefore, it is necessary to round the numbers and record the margins of error to snap the coordinates to a grid system. With the resulting calculation of the target coordinates, it is sufficient to proceed to the next stage, which is **path planning**. The path planning algorithm follows a simple "**Avoid paths that cause conflicts, but if all paths cause conflicts, pause before continuing**" rule. Hence, recording the paths and the timestep the movement will happen is crucial.

For instance, when the intended movement for robot2 and robot3, there is a conflicting movement in *timestep 2* at the *coordinates (3, 3)*. Therefore, robot3 halts for a single timestep before continuing. This is designed so that robots will not take unnecessary detours and create environmental variabilities.

Moreover, The current implementation of the path planning algorithm also takes into account the presence of obstacles presented by the localization and object detection modules as input. The cases for obstacle appearance and avoidance can be divided into two main categories: obstacles appearing at the **turning** location while heading to the destination, and obstacles appearing while heading **straight** towards a destination.

Obstacles appearance during turning can also be considered as the them appearing on a different axis compared to the ongoing movement (robot moving along the x axis, obstacles appear while turning onto the y axis, vice versa). Obstacles encountered this way can be tackled by continuing movement by a single step in the same direction as its ongoing movement, then computing the path as per the usual method. For visualization purposes, this case has been portrayed in Figure 6.2.

### Case 1: Obstacle at Intersection

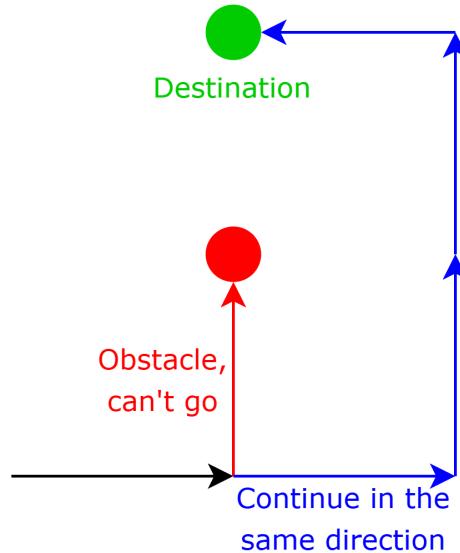


Figure 6.2: Obstacle Avoidance Case 1: Obstacle during Turning

The other case for obstacle appearance can be considered as one where it coincides with the line of movement the robot is currently ongoing (robot moving along the x axis, obstacles appear along that x path). This case can be handled by "going around" the obstacles, appending movements that go around the obstacles both in the positive and negative directions, and continuing the existing path planning algorithm. Figure 6.3 illustrates the case and the proposed solution.

### Case 2: Obstacle along the Path

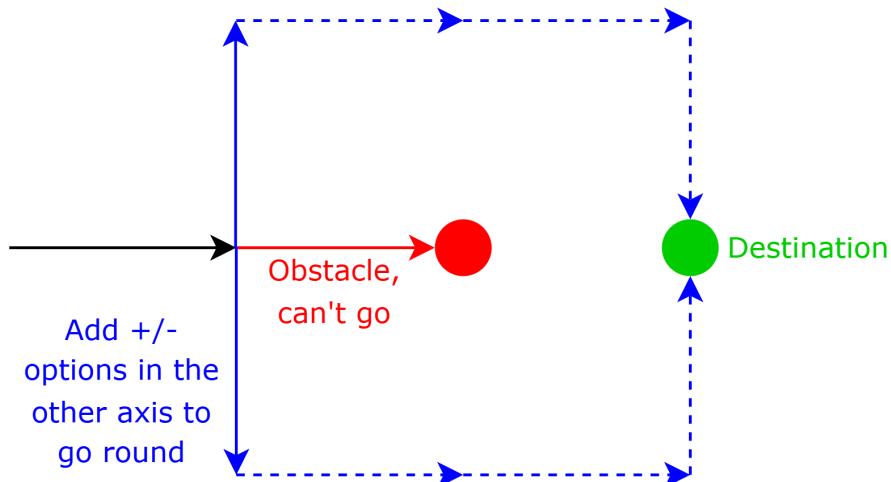


Figure 6.3: Obstacle Avoidance Case 2: Obstacle along the Current Path

Once the appropriate paths have been computed, there are two essential considerations remaining, namely, formatting them to be compatible with communication and motion controller, and covering actions after arriving at the destinations such as orientations of the robots to grip the objects. An illustration of how these two considerations are tackled is displayed in Figure 6.4.

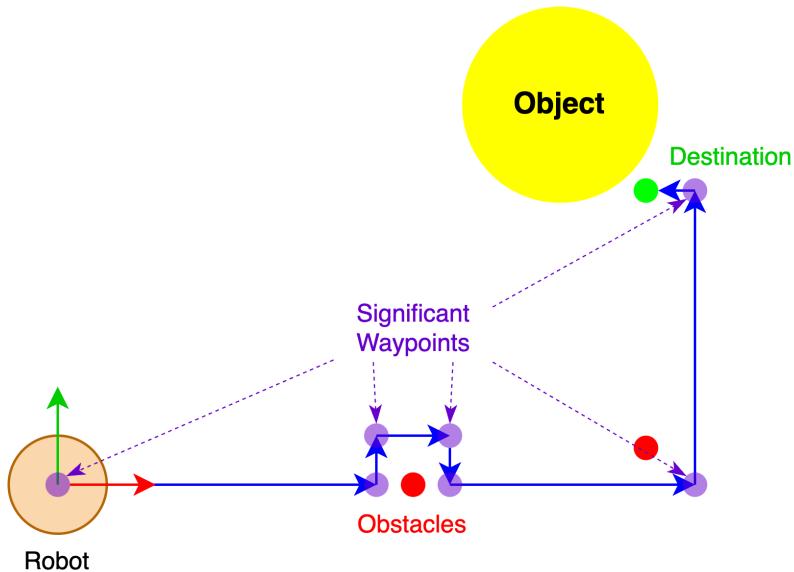


Figure 6.4: Path Waypoint

The computed paths originally lists all coordinates along the paths with a minuscule interval; for long paths, these coordinates can easily reach over a thousand. Considering this with the potential of scalability of swarm systems of numerous robots, this is a glaring weakness. Thus, the paths are translated to discrete waypoints, only recording significant coordinates such as corners for turning, and a repeat of coordinates for

momentarily stopping, in case the robot has to wait for another to move past to avoid colliding. This can translate a thousand-long list of coordinates into potentially several tens of them, dependent on the number of corners the robots have to take, making the structure more efficient.

For appropriate path planning, it is also essential to consider the subsequent actions after robots arrive at their destinations in the equilateral formation around the object. In this case, the orientation of the robots have to be considered. Transmitting the final orientations of the robots will allow them to position themselves with the gripper facing toward the detected object, completing the formation and allowing them to move the object. The swarm can also surround the object in a higher actual radius than the object's, position themselves according to the final orientation, and move in to form a sufficient hold.

# Chapter 7

## Localization

### SLAM Analysis for Robot Navigation

Simultaneous Localization and Mapping (SLAM) is a computational technique used by mobile robots to build a map of an unknown environment while simultaneously estimating their own position within that map. SLAM fuses data from sensors such as LiDAR, IMU, and wheel encoders to incrementally construct a geometric representation of the surrounding environment. By performing scan matching and correcting odometry drift over time, SLAM enables reliable localization and autonomous navigation even in GPS-denied environments. Variants like Graph SLAM use nonlinear optimization to refine pose graphs, while real-time solutions such as SLAM Toolbox offer online mapping and loop closure capabilities.

In our project, SLAM was intended to support three cooperative robots performing collective object transport. We explored both Graph SLAM and SLAM Toolbox as potential solutions. SLAM Toolbox was particularly appealing due to its integration with ROS 2 and support for asynchronous global pose graph optimization. The goal was to construct a shared map, correct odometry drift from both LiDAR (RF2O) and encoder sources, and enable reliable path planning across the arena. SLAM would also assist with multi-robot coordination by aligning their reference frames to a common global map, which is crucial for tasks such as role assignment and distributed transport.

However, SLAM cannot be effectively deployed in our current setup due to a combination of environmental and sensor limitations. The arena's physical dimensions (approximately  $2.1\text{ m} \times 1.4\text{ m}$ ) are too small to allow sufficient motion for loop closures or robust scan alignment. Moreover, the environment lacks distinctive features, providing little geometric structure for scan matching algorithms to latch onto. On the odometry side, wheel encoders on the Dynamixel-based drive system have blind spots, and the RF2O LiDAR odometry exhibits significant drift and instability in low-feature settings. These issues with the odometry are further described below.

## Wheel Odometry

Wheel odometry is solely calculated using the dynamic model of the omnidirectional wheeled robot. Hence, the position or the velocity of the wheel becomes a necessity. However, wheel odometry is only suitable under two conditions: the robot must not have wheel slip, and the encoder must be precise.

The problem that arose during the implementation phase was that the motor (Dynamixel AX-12W) was not able to return the complete position and velocity values. Due to the limited range of the embedded encoder in the motor, the resulting random value is between  $330^\circ$  and  $350^\circ$  (Figure 7.1). Moreover, the value returned by the encoder, according to the documentation, was the percentage of the maximum torque. [58]

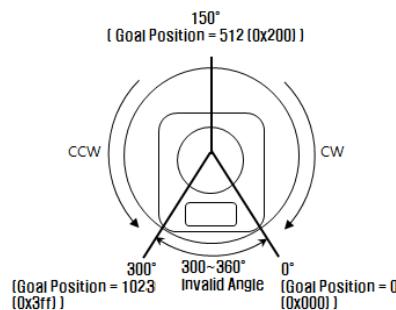


Figure 7.1: Encoder diagram of AX-12W

To work around this, we instead tried the current speed register. However, this approach requires the robot to operate at relatively low speeds to maintain accuracy. Yet, this method will not work during the collective transport phase, as the load to the motor will be different. Hence, LiDAR odometry was considered as the more viable solution for odometry.

## LiDAR Odometry

The LiDAR odometry package that we used was the MAPIRlab rf20 odometer package. The details of the package were described in their paper published in the ICRA conference. The process used for the calculation of LiDAR odometry was based on range flow on velocity estimates.

### 1. Range Flow Constraint

Assuming a static environment, the total derivative of the range measurement  $r(\theta, t)$  at angle  $\theta$  and time  $t$  is zero:

$$\frac{dr(\theta, t)}{dt} = \frac{\partial r}{\partial \theta} \frac{d\theta}{dt} + \frac{\partial r}{\partial t} = 0$$

### 2. Velocity Twist of the Sensor

We represent the robot's planar motion as a twist vector:

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Each point  $(x_i, y_i)$  from the laser scan moves according to:

$$\dot{x}_i = v_x - \omega y_i, \quad \dot{y}_i = v_y + \omega x_i$$

### 3. Range Derivative from Motion

Let  $r_i = \sqrt{x_i^2 + y_i^2}$  be the range. The predicted temporal change in range is:

$$\dot{r}_i = \frac{x_i \dot{x}_i + y_i \dot{y}_i}{r_i}$$

Substituting  $\dot{x}_i$  and  $\dot{y}_i$ :

$$\dot{r}_i = \frac{x_i(v_x - \omega y_i) + y_i(v_y + \omega x_i)}{r_i}$$

$$\dot{r}_i = \frac{v_x x_i + v_y y_i + \omega(x_i y_i - y_i x_i)}{r_i} = \frac{v_x x_i + v_y y_i}{r_i}$$

(The cross-term cancels out.)

### 4. Least-Squares Estimation

Given measured temporal derivatives  $\dot{r}_i^{\text{meas}}$ , the goal is to find  $\mathbf{v}$  that minimizes:

$$E(\mathbf{v}) = \sum_i \left( \dot{r}_i^{\text{meas}} - \dot{r}_i^{\text{pred}} \right)^2$$

This results in a linear system of the form:

$$A\mathbf{v} = b$$

which is solved using robust linear least-squares methods.

## Problems and issues

In our experiments with RF2O-based LiDAR odometry, we encountered several critical issues. Firstly, the robot's orientation appeared inconsistent over time, with the estimated  $x$  and  $y$  positions occasionally swapping, leading to significant localization errors. Secondly, the system continued to update the pose incrementally even when the robot was completely stationary, indicating motion when none occurred. Lastly, during actual movement, the estimated pose exhibited significant drift from the ground truth trajectory, compromising the accuracy of the robot's localization.

The root causes of these problems can be traced to inherent limitations of RF2O. The orientation inconsistencies are due to the absence of an absolute reference frame—RF2O cannot distinguish between rotated coordinate frames without external input, leading to ambiguity when the robot turns. The false motion during stationarity arises from sensor noise and small variations in consecutive scans, which RF2O interprets as real movement. The drift during motion is a result of accumulating small scan alignment errors, especially in low-feature environments, due to the purely incremental nature of RF2O's pose estimation. To further asses the issue we decided to validate LiDAR odometry using an overhead camera.

## Odometry Validation

The performance evaluation of the LiDAR odometry is carried out using an overhead camera setup. To ensure that our odometry is accurate enough for use in SLAM, the camera tracks the robot's movement and provides a ground-truth odometry reference. Then, the data from the camera are compared to the LiDAR's.

The testing was carried out in a controlled indoor environment using a Logitech C922 PRO HD Stream webcam mounted on a studio stand (Figure 7.2). The tests were carried out on a flat surface free from slipperiness and reflections(Figure 7.3).



Figure 7.2: Camera setup



Figure 7.3: Arena setup

The Augmented Reality University of Cordoba (ArUco) markers are used to locate the robot's position within the testing arena. DICT\_4x4\_50 ArUco markers were selected for this testing because they're the least complex markers, making computation faster and more reliable for real-time detection.

The ArUco markers were placed directly above the robot hardware, while the camera faced directly into the arena (Figure 7.4). After that, our overhead camera system

extracted the ArUcos from each camera frame and calculated the actual position from the map using the Perspective-n-Point (PnP) pose computation.

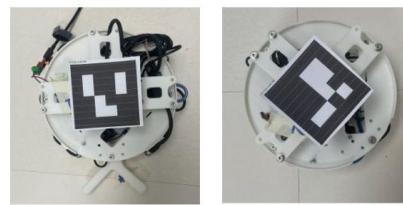


Figure 7.4: ArUco markers on the swarms

## ArUco Marker Localization

Given the four detected corner points of an ArUco marker in image coordinates:

$$\text{Corners} = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\} = \{\text{topLeft}, \text{topRight}, \text{bottomRight}, \text{bottomLeft}\}$$

### 1. Center of the Marker

$$c_x = \frac{x_{\text{topLeft}} + x_{\text{bottomRight}}}{2}, \quad c_y = \frac{y_{\text{topLeft}} + y_{\text{bottomRight}}}{2}$$

### 2. Direction Vector of the Top Edge

$$\Delta x = x_{\text{topRight}} - x_{\text{topLeft}}, \quad \Delta y = y_{\text{topRight}} - y_{\text{topLeft}}$$

### 3. Orientation Estimation

Display orientation in degrees:

$$\theta_{\text{display}} = \arctan 2(\Delta y, -\Delta x)$$

### 4. Normalized Image Coordinates

Assuming image width  $W$  and height  $H$ :

$$x_{\text{norm}} = \frac{c_x}{W}, \quad y_{\text{norm}} = \frac{c_y}{H}$$

### 5. Real-World Position Mapping

Given real-world map dimensions  $S_x$  and  $S_y$ :

$$x_{\text{map}} = (x_{\text{norm}} - 0.5) \cdot S_x, \quad y_{\text{map}} = -(y_{\text{norm}} - 0.5) \cdot S_y$$

The given formulation allows the transformation of detected 2D marker positions into real-world coordinates as figure shown:

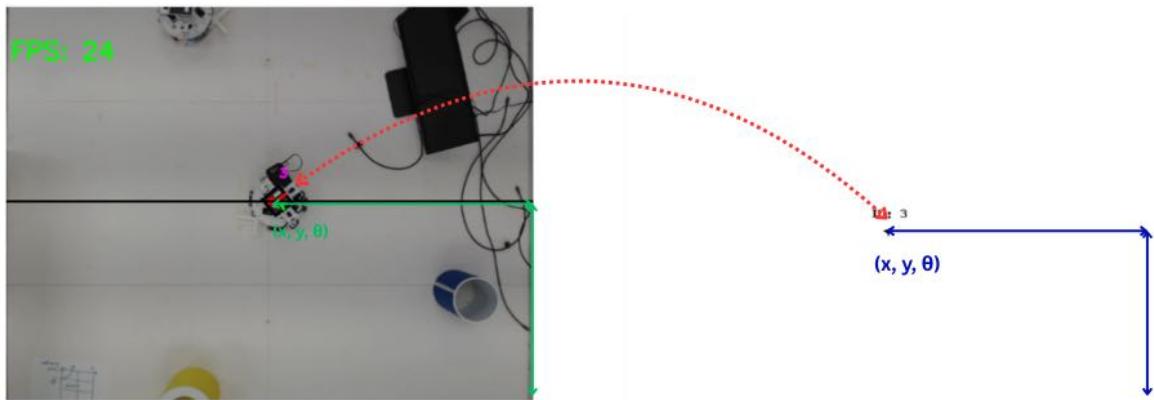


Figure 7.5: Visualization of real-time mapping of swarms using OpenCV and Turtle

After obtaining data from both the camera and LiDAR, we evaluated the trajectory error produced by the LiDAR odometry through the following tests:

1. **Translational motion** (holonomic): The robot moved 50 cm in the positive  $x$ -direction via sideways motion ( $45^\circ$  to the traveling direction).
2. **Translational motion** (non-holonomic): The robot moved 50 cm in the positive  $x$ -direction via forward motion ( $0^\circ$  to the traveling direction).
3. **Rotational motion**: The robot performed a  $360^\circ$  turn clockwise and counterclockwise in place.

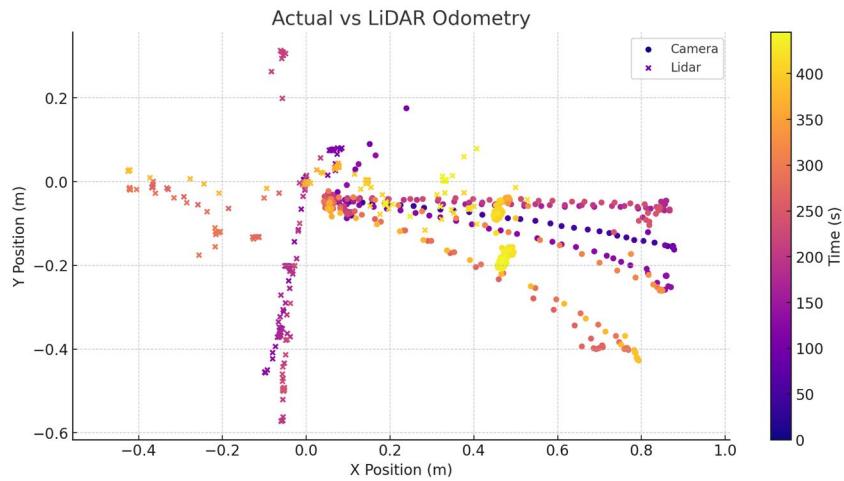


Figure 7.6: Visualization of robot motion based on LiDAR and camera odometry in a 2D plane.

As shown in Figure 7.6, the robot's ability to move in a straight line significantly deteriorated when switching from moving at a  $45^\circ$  angle to the direction of travel (using two wheels) to moving in the direction it is facing (using all four wheels). The goal of this analysis was to evaluate whether the LiDAR odometry could accurately capture trajectory errors and be reliably used for SLAM or feedback control.

However, one issue encountered was that LiDAR odometry assigned a different  $x$ -axis orientation for each test. To address this, we compensated for the discrepancy by remapping the  $x$ -axis to match the robot's initial direction of motion during error calculation.

Figure 8.7 presents the detailed numerical comparison of the robot's estimated position and orientation ( $x$ ,  $y$ , and  $\theta$ ) from both the LiDAR and camera odometry across all test scenarios.

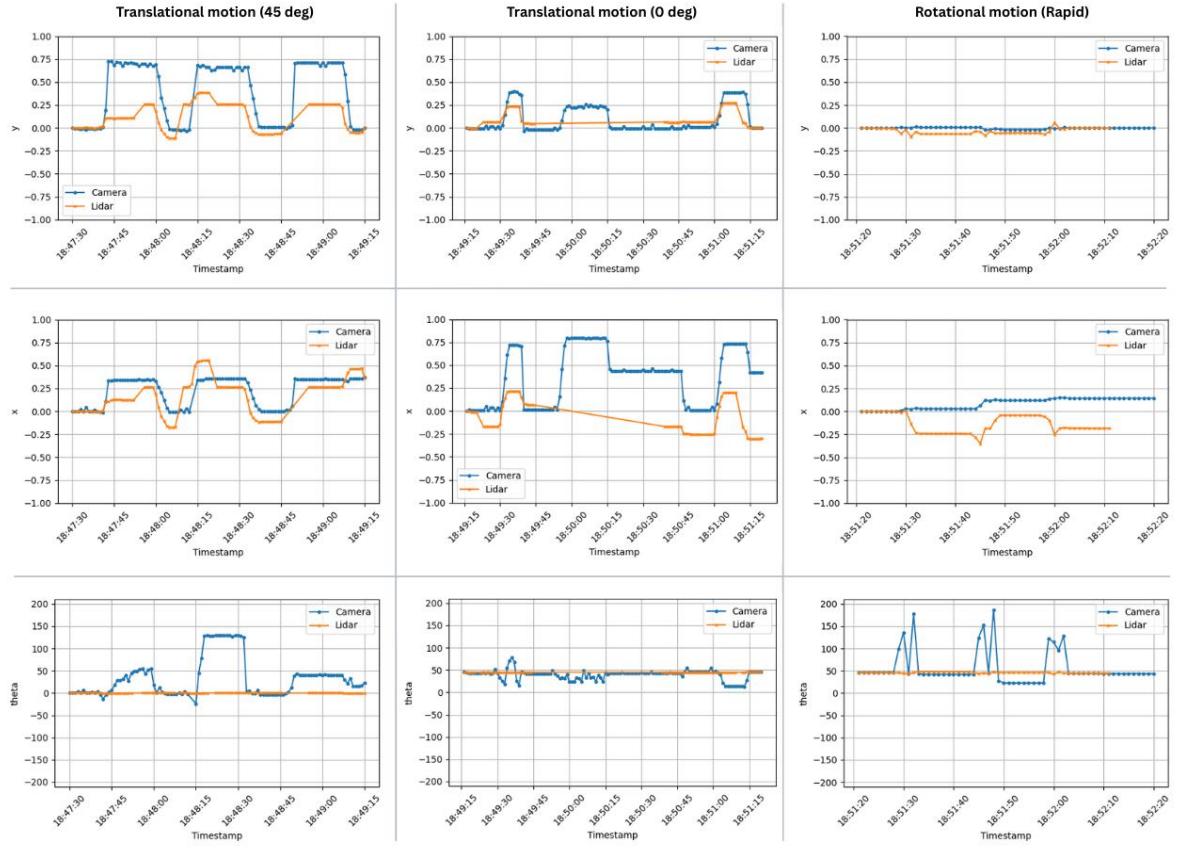


Figure 7.7: Detailed Visualization of LiDAR vs actual odometry

Metric	Translational (45°)	Translational (0°)	Rotational
NRMSE (Y)	0.483	0.597	1.282
NRMSE (X)	0.389	0.614	0.768
NRMSE ( $\theta$ )	0.360	0.629	0.407

Table 7.1: Normalised Root Mean Squared Error (NRMSE) Comparison Between Camera and LiDAR

From our experiments, we observed that the current LiDAR odometry often deviates from the actual trajectory, especially in movements that involve rotation or sliding. These inconsistencies show that the LiDAR setup may not provide sufficient accuracy for precise localization tasks or feedback data. As a result, we propose using the overhead camera system with ArUco markers as a temporary but reliable substitute for ground-truth positioning. This approach offers improved consistency and will serve as our main reference until a more robust odometry solution is implemented.

# Chapter 8

## Result

This chapter details the result of our project through images of our demonstration achieving our objectives.

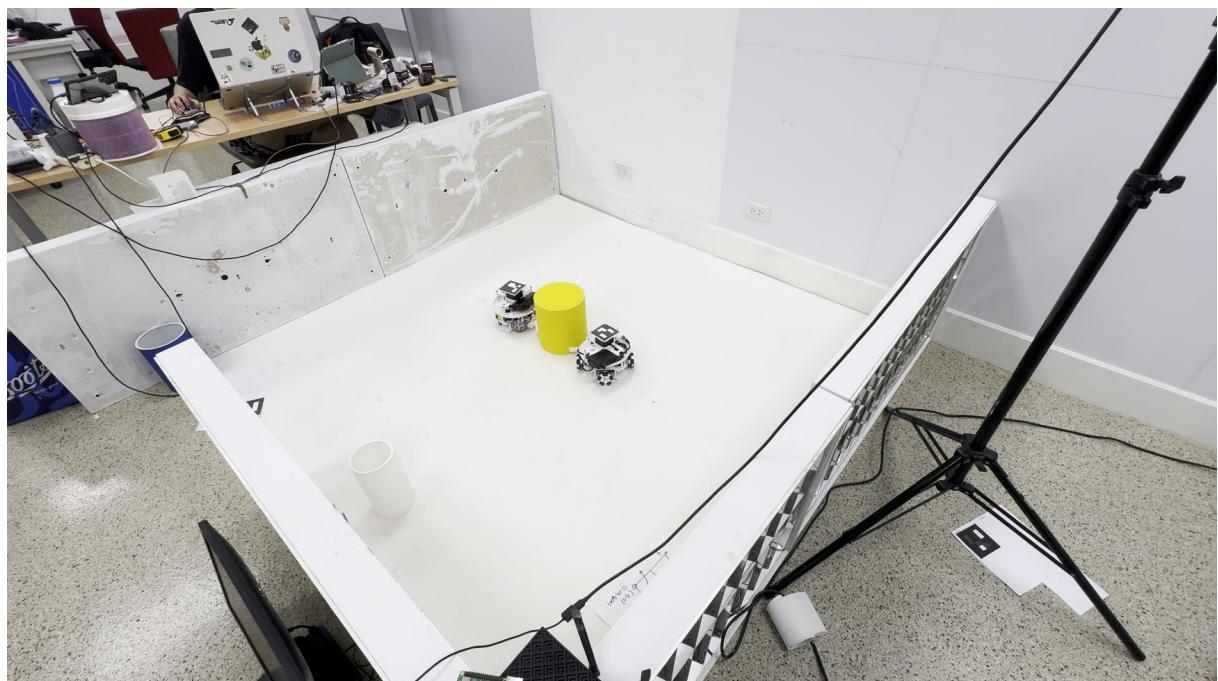


Figure 8.1: Robot moving towards object

The image above shows the robots successfully moving towards the object in a synchronized manner.

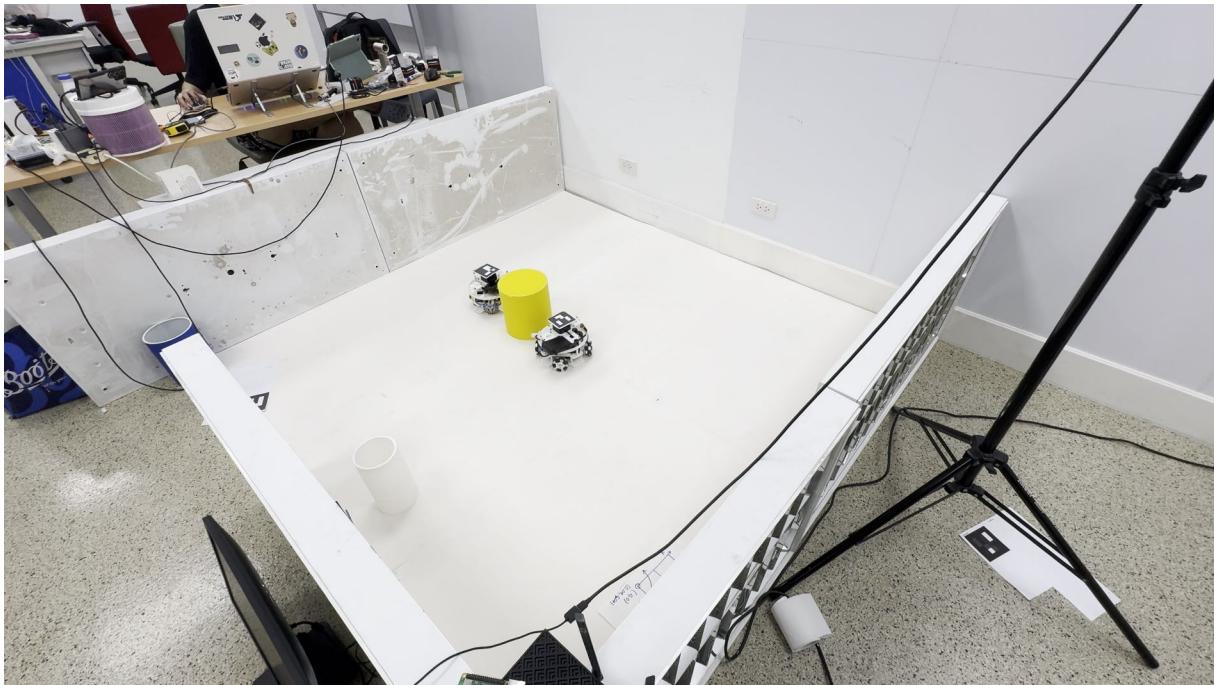


Figure 8.2: Robots grip and transport object

The image shows how the robots grip the object at the correct position. The robots then transports the object collectively in the global y direction

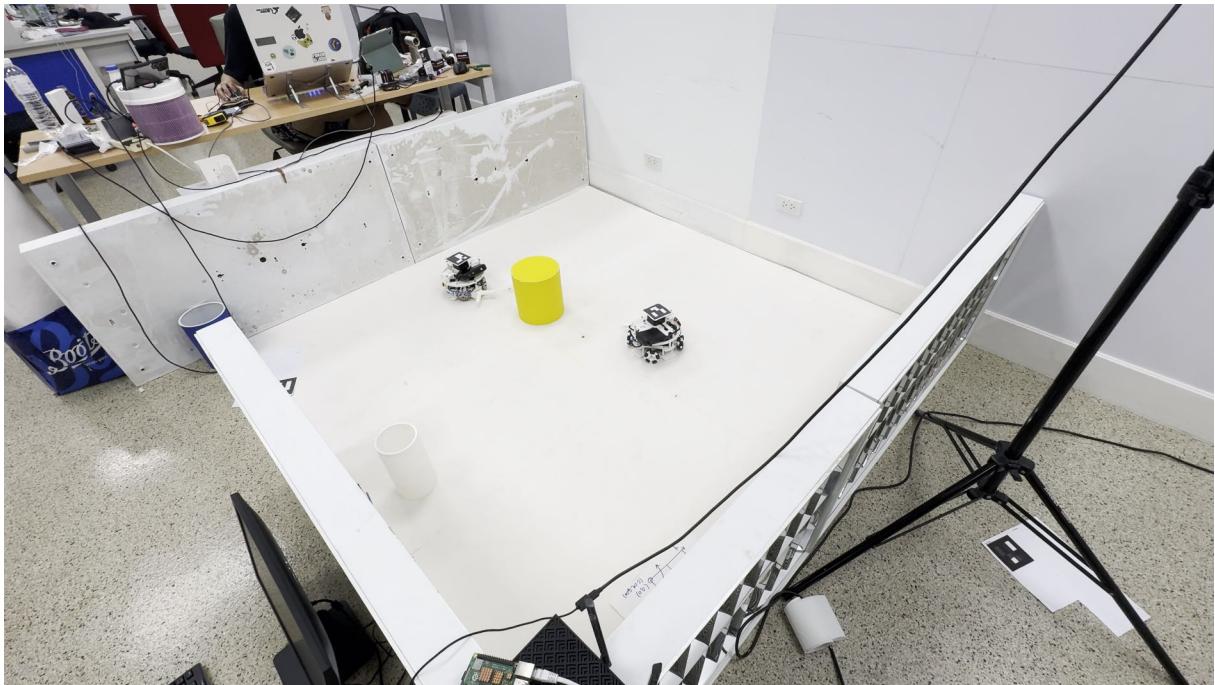


Figure 8.3: Robots return to Home position

After transporting the object the robots then move towards their home positions



Figure 8.4: Robot detecting object

The above image shows how the robot detects an object by creating the box around it

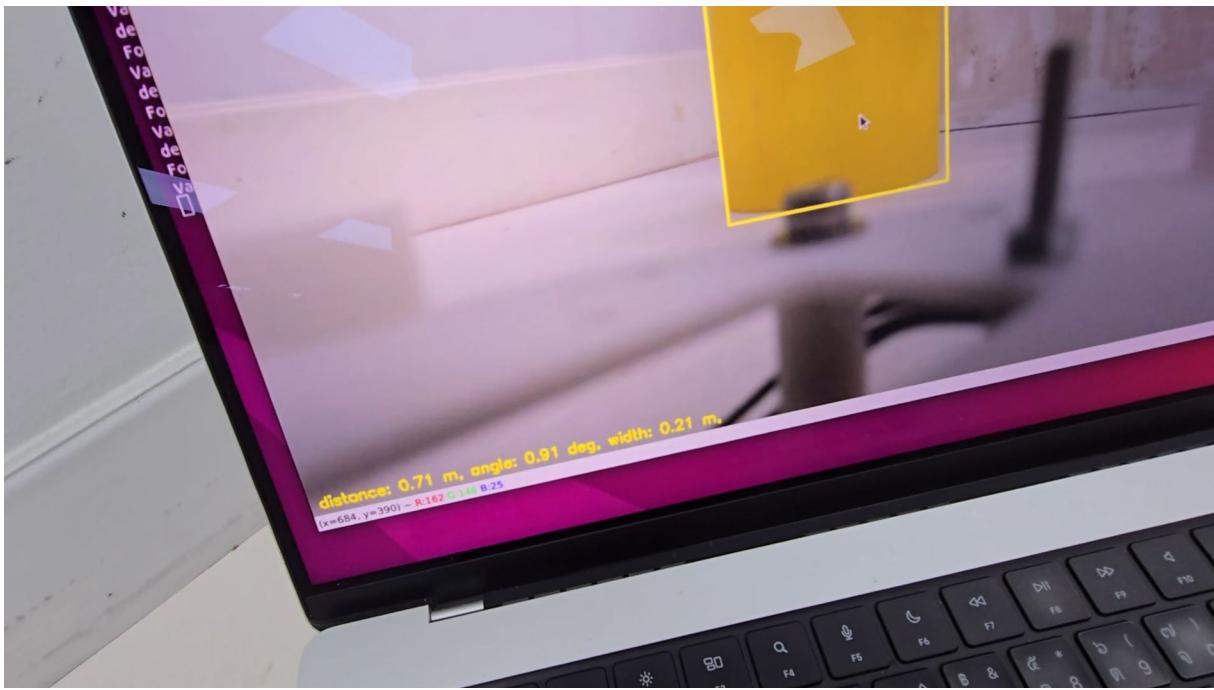


Figure 8.5: Measurements of the object

The object detection module then processes the measurements necessary. The distance between the object and the robot, the angle between the object and the robot, and the width of the object.



Figure 8.6: Accuracy of the measurements

The actual measurement is only off by around 5 cm.

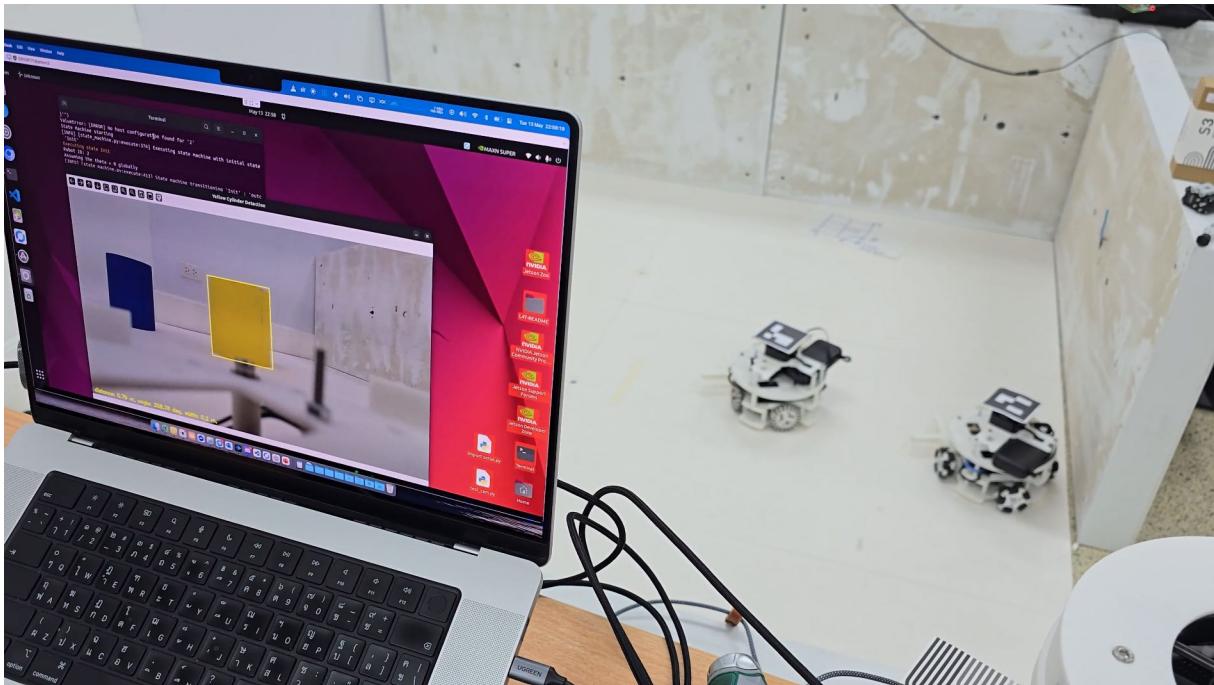


Figure 8.7: Robots navigating to object after detecting it

The robots after detecting the object then navigate towards it.

# Chapter 9

## Conclusion

Swarm communication has been successfully implemented using a decentralized peer-to-peer architecture facilitated by socket communication. This approach was selected for its low latency and reliability, both of which are essential for real-time data exchange among robots. A three-way handshake mechanism was incorporated to ensure message delivery integrity, along with a consensus algorithm to resolve simultaneous object detection and taskmaster assignment conflicts. With coordinate streaming, dynamic path planning, and task execution now fully integrated, the communication system provides a solid foundation for coordinated swarm behavior, enabling seamless transitions between detection, planning, and movement phases.

Additionally, the object detection system utilizing multi-sensor fusion of the camera and S3 RPLiDAR accurately measures relative distance, relative angle, and estimated width, aided by the variation factor function. This function yields an error margin of 1 centimeter for cylinders with diameters of 16 and 20 centimeters. The model was also tested with objects of different sizes, specifically a cylinder with a diameter of 12 centimeters, resulting in a slightly higher error margin of 2 centimeters. Overall, the object detection component was completed within the expected timeline.

During the implementation stage, we noticed constraints on the Dynamixel AX-12W motor encoder. This experience showed the importance of thoroughly reviewing hardware documentation prior to system integration. Misinterpretation of sensor capabilities can significantly reduce system performance and reliability. However, we created an overhead camera system with ArUco markers, which the system effectively extracted. In our tests, this method exceeded LiDAR in terms of consistency and accuracy. This case displays the value of flexible problem solving. Using ArUco-based vision as a temporary but effective solution, we were able to deliver accurate ground-truth positioning during the testing and evaluation phases.

On the hardware side we successfully transitioned from a CAD model into a functional physical prototype. Through iterative development, components such as the base frame, shaft supports, and custom sensor mounts were prototyped using PLA+ and precision-calibrated with tools like vernier calipers. The use of a Bambu Lab P1S printer allowed high-accuracy press-fit assemblies, particularly for bearing integration and shaft alignment.

The initial drive system was built around Dynamixel AX-12W motors, which provided modular control but suffered from limitations in encoder accuracy—posing challenges for precise odometry. To address this, work has begun on a redesigned base platform using DC motors with optical encoders for improved feedback, driven by an ESP32 microcontroller. While this new platform is not yet fully integrated, it demonstrates a forward-looking shift toward higher control fidelity and better hardware-software integration.

In summary, this project successfully demonstrates the core principles of decentralized swarm robotics, providing a solid foundation for future enhancements in control, coordination, and scalability.

## **Appendix A**

### **Object Detection Model Testing Result**

Table A.1: Predicted diameter under different lighting conditions of 20 cm diameter cylinder, placed at actual relative angle of 0 degree.

<b>Lighting Condition</b>	<b>Distance (m)</b>	<b>Angle (°)</b>	<b>Predicted Width (m)</b>
1	0.5	0.21	0.20
2	0.5	0.21	0.20
3	0.5	0.21	0.20
4	0.5	0.21	0.20
5	0.5	0.28	0.20
6	0.5	0.28	0.20
7	0.5	0.13	0.20
8	0.5	0.21	0.20
9	0.5	0.21	0.21
<hr/>			
1	1.00	0.37	0.20
2	1.01	0.43	0.20
3	1.00	0.43	0.20
4	1.00	0.37	0.20
5	1.00	0.43	0.20
6	1.00	0.31	0.20
7	1.00	0.29	0.20
8	1.00	0.33	0.20
9	1.00	0.33	0.20
<hr/>			
1	2.00	0.18	0.20
2	1.99	0.18	0.20
3	2.00	0.18	0.20
4	2.00	0.18	0.20
5	2.00	0.18	0.20
6	1.99	0.18	0.20
7	2.00	0.18	0.20
8	2.00	0.18	0.20
9	2.00	0.18	0.20
<hr/>			
1	3.00	0.35	0.20
2	3.00	0.29	0.20
3	3.00	0.29	0.20
4	2.99	0.29	0.20
5	3.00	0.29	0.20
6	3.00	0.29	0.20
7	3.01	0.35	0.19
8	3.00	0.29	0.20
9	3.00	0.29	0.20

Table A.2: Predicted diameter under different lighting conditions of 16 cm diameter cylinder, placed at actual relative angle of 0 degree.

<b>Lighting Condition</b>	<b>Distance (m)</b>	<b>Angle (°)</b>	<b>Predicted Width (m)</b>
1	0.5	358.11	0.16
2	0.5	358.11	0.16
3	0.5	358.11	0.16
4	0.5	358.05	0.16
5	0.5	358.11	0.16
6	0.5	358.11	0.16
7	0.5	358.11	0.16
8	0.5	358.11	0.16
9	0.5	358.11	0.16
1	1.00	0.18	0.16
2	1.00	0.18	0.16
3	1.00	0.18	0.16
4	1.00	0.18	0.16
5	1.00	0.12	0.16
6	1.00	0.18	0.16
7	1.00	0.12	0.16
8	1.00	0.18	0.16
9	1.00	0.18	0.16
1	2.00	0.55	0.16
2	2.00	0.61	0.16
3	2.00	0.61	0.16
4	2.00	0.61	0.16
5	2.00	0.55	0.16
6	2.00	0.55	0.16
7	2.00	0.55	0.16
8	2.00	0.55	0.16
9	2.00	0.55	0.16
1	3.00	0.22	0.16
2	3.01	0.28	0.16
3	3.00	0.28	0.16
4	3.00	0.22	0.17
5	3.00	0.28	0.16
6	3.00	0.28	0.16
7	3.00	0.28	0.17
8	3.00	0.28	0.16
9	3.00	0.28	0.16

Table A.3: Predicted diameter under different lighting conditions of 12 cm diameter cylinder, placed at actual relative angles of 2, 0, 1.5, and 0 degrees.

<b>Lighting Condition</b>	<b>Distance (m)</b>	<b>Angle (°)</b>	<b>Predicted Width (m)</b>
1	0.5	2.01	0.13
2	0.5	2.01	0.13
3	0.5	2.01	0.12
4	0.5	2.00	0.14
5	0.5	2.01	0.13
6	0.5	2.01	0.12
7	0.5	2.02	0.13
8	0.5	2.01	0.14
9	0.5	2.01	0.13
1	1.00	0.21	0.12
2	1.00	0.21	0.13
3	1.00	0.20	0.13
4	1.00	0.21	0.12
5	1.00	0.21	0.12
6	1.01	0.23	0.14
7	1.00	0.21	0.12
8	1.00	0.23	0.13
9	1.00	0.21	0.13
1	2.00	1.65	0.12
2	2.00	1.65	0.12
3	2.01	1.68	0.12
4	2.00	1.65	0.13
5	2.00	1.68	0.13
6	2.00	1.65	0.12
7	2.00	1.65	0.12
8	1.99	1.68	0.14
9	2.00	1.68	0.14
1	3.00	0.05	0.12
2	3.00	0.05	0.12
3	3.00	0.05	0.14
4	3.01	0.05	0.12
5	3.00	0.05	0.14
6	3.00	0.05	0.13
7	3.00	0.05	0.12
8	3.00	0.05	0.13
9	3.00	0.05	0.12

# Bibliography

- [1] L. Türkler, T. Akkan, and L. Ö. Akkan, "Usage of evolutionary algorithms in swarm robotics and design problems", *Sensors*, vol. 22, no. 12, p. 4437, 2022. DOI: 10.3390/s22124437.
- [2] S. Das, "Bio-inspired communication strategies in swarm robotics", in *Advances in Computational Intelligence and Robotics*. 2024, pp. 77–100. DOI: 10.4018/979-8-3693-1277-3.ch006.
- [3] R. Ibrahim, M. Alkilabi, A. R. H. Khayeat, and E. Tuci, "Enhancing robustness of swarm robotics systems in a perceptual discrimination task", *International Journal of Computing and Digital Systems*, vol. 15, no. 1, pp. 1213–1222, 2024. DOI: 10.12785/ijcds/160189.
- [4] A. Ayari and S. Bouamama, "Evolutionary swarm robotics: A methodological approach for task and path planning", in *2023 IEEE Afro-Mediterranean Conference on Artificial Intelligence (AMCAI)*, 2023. DOI: 10.1109/amcai59331.2023.10431514.
- [5] S. M. Perera, R. J. Myers, K. Sullivan, K. Byassee, H. Song, and A. Madanayake, "Integrating communication and sensor arrays to model and navigate autonomous unmanned aerial systems", *Electronics*, vol. 11, no. 19, p. 3023, 2022. DOI: 10.3390/electronics11193023.
- [6] S. S. R., S. Mohanty, and D. S. Elias, "Control and coordination of a swarm of unmanned surface vehicles using deep reinforcement learning in ros", *arXiv (Cornell University)*, 2023. DOI: 10.48550/arxiv.2304.08189.
- [7] Z. Wang, J. Li, J. Li, and C. Liu, "A decentralized decision-making algorithm of uav swarm with information fusion strategy", *Expert Systems With Applications*, vol. 237, p. 121 444, 2024. DOI: 10.1016/j.eswa.2023.121444.
- [8] M. Yasser, O. Shalash, and O. Ismail, "Optimized decentralized swarm communication algorithms for efficient task allocation and power consumption in swarm robotics", *Robotics*, vol. 13, no. 5, p. 66, 2024. DOI: 10.3390/robotics13050066.
- [9] N. Nedjah, L. M. Ribeiro, and L. De Macedo Mourelle, "Communication optimization for efficient dynamic task allocation in swarm robotics", *Applied Soft Computing*, vol. 105, p. 107 297, 2021. DOI: 10.1016/j.asoc.2021.107297.
- [10] E. Beck, B. Shin, S. Wang, T. Wiedemann, D. Shutin, and A. Dekorsy, "Swarm exploration and communications: A first step towards mutually-aware integration by probabilistic learning", *Electronics*, vol. 12, no. 8, p. 1908, 2023. DOI: 10.3390/electronics12081908.

- [11] S. Zhang, E. Staudinger, R. Pohlmann, and A. Dammann, "Cooperative communication, localization, sensing and control for autonomous robotic networks", in *2021 IEEE International Conference on Autonomous Systems (ICAS)*, 2021. DOI: 10.1109/icas49788.2021.9551201.
- [12] V. Patel, V. Mehta, M. Bolic, and I. Mantegh, "A hybrid framework for object distance estimation using a monocular camera", in *Proceedings of the IEEE International Conference (or other conference details)*, IEEE, 2023. DOI: 10.1109/10311189.
- [13] J. Janai, F. Güney, A. B, and A. G, "Computer vision for autonomous vehicles: Problems, datasets and state of the art", *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020. DOI: 10.1561/0600000079.
- [14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. DOI: 10.1023/B:VISI.0000029664.99615.94.
- [15] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177.
- [16] A. Sachan, "Object detection using deep learning: Faster r-cnn, yolo, ssd", *Data Science Central*, 2019. [Online]. Available: <https://www.datasciencecentral.com/zero-to-hero-guide-to-object-detection-using-deep-learning-faster/>.
- [17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2016, pp. 779–788, 2016. DOI: 10.1109/CVPR.2016.91. [Online]. Available: <https://arxiv.org/abs/1506.02640>.
- [18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks", in *Advances in Neural Information Processing Systems (NIPS)*, 2015. DOI: 10.48550/arXiv.1506.01497.
- [19] W. Liu et al., "Ssd: Single shot multibox detector", in *Computer Vision – ECCV 2016. Lecture Notes in Computer Science*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9905, Springer, Cham, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0\_2.
- [20] V. K. Kaliappan, R. Thangaraj, P. Pandiyan, K. Mohanasundaram, S. Anandamurugan, and D. Min, "Real-time face mask position recognition system using yolo models for preventing covid-19 disease spread in public places", *International Journal of Adaptive and Autonomous Systems*, vol. 34, no. 1, pp. 73–82, 2023. DOI: 10.1504/IJAHUC.2023.128499.
- [21] V. K. Kaliappan, M. S. Shanmugasundaram, L. Ravikumar, and G. B. Hiremath, "Performance analysis of yolov8, rcnn, and ssd object detection models for precision poultry farming management", in *2023 IEEE 3rd International Conference on Applied Electromagnetics, Signal Processing, & Communication (AESPC)*, Bhubaneswar, India: IEEE, 2023, pp. 1–6. DOI: 10.1109/AESPC59761.2023.10389906.

- [22] A. Geiger, P. Lenz, and R. Urtasun, “Vision meets robotics: The kitti dataset”, *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013. DOI: 10.1177/0278364913491297. [Online]. Available: <https://arxiv.org/abs/1910.05395>.
- [23] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017. DOI: 10.1109/CVPR.2017.16.
- [24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”, *Advances in Neural Information Processing Systems (NIPS)*, pp. 5099–5108, 2017. DOI: 10.48550/arXiv.1706.02413.
- [25] Y. Zhou, O. Tuzel, V. Le, and V. Koltun, “Voxelnet: End-to-end learning for point cloud-based 3d object detection”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4490–4499. DOI: 10.1109/CVPR.2018.00472.
- [26] Y. Yan, Y. Mao, and B. Li, “Second: Sparsely embedded convolutional detection”, in *Sensors*, vol. 18, 2018, p. 3337. DOI: 10.3390/s18103337.
- [27] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum pointnets for 3d object detection from rgbd data”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 918–927. DOI: 10.1109/CVPR.2018.00102.
- [28] J. Ku, M. Pon, and C. C. Fowlkes, “Multi-view 3d object detection network for autonomous driving”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3676–3684, 2018. DOI: 10.1109/CVPR.2018.00387.
- [29] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving”, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1907–1915. DOI: 10.1109/CVPR.2017.206.
- [30] R. Paul, S. Bag, D. Roy, and M. Nasipuri, “Object detection and pose estimation from rgbd and depth data for real-time robotic applications”, in *Proceedings of the International Conference on Pattern Recognition and Machine Intelligence*, Springer, 2021, pp. 95–102. DOI: 10.1007/978-3-030-71051-4\_10.
- [31] Z. Yuan, Y. Zhang, M. Li, and W. Li, “Accurate covariance estimation for pose data from iterative closest point algorithms”, *NAVIGATION: Journal of the Institute of Navigation*, vol. 70, no. 2, pp. 383–398, 2023. DOI: 10.1002/navi.562.
- [32] A. Kendall, M. Grimes, and R. Cipolla, “Posenet: A convolutional network for real-time 6-dof camera relocalization”, in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2938–2946. DOI: 10.1109/ICCV.2015.336.
- [33] S. Peng, Y. Liu, Q. Huang, H. Bao, and X. Zhou, “Pvnet: Pixel-wise voting network for 6dof pose estimation”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4561–4570. DOI: 10.1109/CVPR.2019.00469.

- [34] B. Chen, G. Wang, X. Liu, M. Wang, Y. Wang, and Y. Liu, "Occlusion-robust object pose estimation with holistic representation", in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2929–2938. DOI: 10.1109/WACV51458.2022.00300.
- [35] A. Barbadekar, S. Raut, R. Gaikwad, T. Gadad, S. Ghulaxe, and N. Dhabale, "Exploring enhanced localization techniques: Lidar-slam for mobile robots", in *2023 Global Conference on Internet of Things and Cyber-Physical Systems (GCITC)*, 2023. DOI: 10.1109/gcitc60406.2023.10426008.
- [36] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i", *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. DOI: 10.1109/mra.2006.1638022.
- [37] T. Chong, X. Tang, C. Leng, M. Yogeswaran, O. Ng, and Y. Chong, "Sensor technologies and simultaneous localization and mapping (slam)", *Procedia Computer Science*, vol. 76, pp. 174–179, 2015. DOI: 10.1016/j.procs.2015.12.336.
- [38] B. Udugama, "Evolution of slam: Toward the robust-perception of autonomy", in *arXiv preprint*, Available at: <https://doi.org/10.48550/arXiv.2302.06365>, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2302.06365>.
- [39] A. Sahoo, S. K. Dwivedy, and P. Robi, "Advancements in the field of autonomous underwater vehicles", in *Ocean Engineering*, vol. 181, Elsevier, 2019, pp. 145–160. DOI: 10.1016/j.oceaneng.2019.04.011. [Online]. Available: <https://doi.org/10.1016/j.oceaneng.2019.04.011>.
- [40] Y. Bisheng, L. Fuxun, and H. Ronggang, "Progress, challenges, and perspectives of 3d lidar point cloud processing", *DOAJ (Directory of Open Access Journals)*, 2017. DOI: 10.11947/j.agcs.2017.20170351.
- [41] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age", *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016. DOI: 10.1109/tro.2016.2624754.
- [42] B. Langmann, K. Hartmann, and O. Loffeld, "Depth camera technology comparison and performance evaluation", in *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, Barcelona, Spain: SciTePress, 2012, pp. 438–444. [Online]. Available: <https://doi.org/10.5220/0003778304380444>.
- [43] C. Peng, "Depth camera point cloud sharpening", in *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE)*, Taipei, Taiwan: IEEE, 2023, pp. 102–106. DOI: 10.1109/icce-taiwan58799.2023.10226686.
- [44] K. Huang, S. Zhang, J. Zhang, and D. Tao, "Event-based simultaneous localization and mapping: A comprehensive survey", *arXiv*, 2023. DOI: 10.48550/arxiv.2304.09793.
- [45] Y. Li, J. Su, L. Liu, and P. Liu, "Object detection based on the fusion of sparse lidar point cloud and dense stereo pseudo point cloud", in *IEEE*, Jan. 2024. DOI: 10.1109/NNICE61279.2024.10498214.
- [46] X. Gu, X. Wang, and Y. Guo, "A review of research on point cloud registration methods", *IOP Conference Series Materials Science and Engineering*, vol. 782, no. 2, p. 022070, 2020. DOI: 10.1088/1757-899x/782/2/022070.

- [47] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time", in *Proceedings of Robotics: Science and Systems (RSS)*, 2014. [Online]. Available: <https://doi.org/10.15607/rss.2014.x.007>.
- [48] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam", *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 4, pp. 31–43, 2010.
- [49] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization", in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.
- [50] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter", *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021. DOI: 10.1109/lra.2021.3064227.
- [51] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam", in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.
- [52] S. Macenski, *Slam toolbox for lifelong mapping and localization*, ROSCon 2021 Presentation, 2021.
- [53] A. Mehta and A. Modi, "Robust sliding mode protocols for formation of quadcopter swarm", 2024.
- [54] F. Martinez and A. Rendon, "A swarm-based flocking control algorithm for exploration and coverage of unknown environments", 2023.
- [55] E. Bonabeau, A. Sobkowski, G. Theraulaz, and J. L. Deneubourg, "Adaptive task allocation inspired by a model of division of labour in social insects", in *Bio-computing and Emergent Computation: Proceedings of BCEC97*, London, UK: World Scientific, 1997, pp. 36–45.
- [56] M. Brambilla, C. Pinciroli, M. Birattari, and M. Dorigo, "Property-driven design for swarm robotics", in *Proceedings of 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Richland: IFAAMAS, 2012, pp. 139–146.
- [57] D. St-Onge, J. Le Ny, and G. Beltrame, "Swarm-slam: Sparse decentralised collaborative simultaneous localization and mapping framework for multi-robot systems", *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1720–1727, 2023.
- [58] ROBOTIS Co., Ltd., *AX-12W: Robotis dynamixel actuator technical documentation*, Accessed: 2025-05-14, 2024. [Online]. Available: <https://emanual.robotis.com/docs/en/dxl/ax/ax-12w/>.
- [59] L. L. Bayindir and E. Şahin, "A review of studies in swarm robotics", *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 15, no. 2, pp. 115–126, 2007. [Online]. Available: <https://journals.tubitak.gov.tr/elektrik/vol15/iss2/2/>.
- [60] A. R. Cheraghi, S. Shahzad, and K. Graffi, "Past, present, and future of swarm robotics", in *Lecture Notes in Networks and Systems*. 2021, pp. 190–233. DOI: 10.1007/978-3-030-82199-9\_13.

- [61] S. E. Ghazouali, Y. Mhirit, A. Oukhrid, U. Michelucci, and H. Nouira, "Fusionvision: A comprehensive approach of 3d object reconstruction and segmentation from rgb-d cameras using yolo and fast segment anything", *Sensors*, vol. 24, no. 9, p. 2889, 2024. DOI: 10.3390/s24092889.
- [62] M. Kegeleirs, G. Grisetti, and M. Birattari, "Swarm slam: Challenges and perspectives", *Frontiers in Robotics and AI*, vol. 8, 2021. DOI: 10.3389/frobt.2021.618268.
- [63] Y. Koifman, A. Barel, and A. M. Bruckstein, "Distributed and decentralized control and task allocation for flexible swarms", *arXiv*, 2024. DOI: 10.48550/arxiv.2405.13941.
- [64] N. Nedjah and L. J. Silva, "Review of methodologies and tasks in swarm robotics towards standardization", *Swarm and Evolutionary Computation*, vol. 50, p. 100565, 2019. DOI: 10.1016/j.swevo.2019.100565.
- [65] J. Qian, Y. Lv, Y. Gao, and J. Wang, "Frustum-ldgcn: A high efficient 3d object detection network with frustum proposal", in *2023 IEEE International Conference on Automation Science and Engineering (CASE)*, 2023. DOI: 10.1109/case56687.2023.10260350.
- [66] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application", in *Lecture Notes in Computer Science*, 2005, pp. 10–20. DOI: 10.1007/978-3-540-30552-1\_2. [Online]. Available: [https://doi.org/10.1007/978-3-540-30552-1\\_2](https://doi.org/10.1007/978-3-540-30552-1_2).
- [67] R. Benkis et al., "A survey and practical application of slam algorithms", in *2024 Progress In Electromagnetics Research Symposium (PIERS)*, 2024, pp. 1–10. DOI: 10.1109/piers62282.2024.10618240.
- [68] W. Chen et al., "Overview of multi-robot collaborative slam from the perspective of data fusion", *Machines*, vol. 11, no. 6, p. 653, 2023. DOI: 10.3390/machines11060653.
- [69] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018. DOI: 10.1109/tpami.2017.2658577.
- [70] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system", *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015. DOI: 10.1109/tro.2015.2463671.
- [71] X. Liu et al., "Slideslam: Sparse, lightweight, decentralized metric-semantic slam for multi-robot navigation", *arXiv*, 2024. [Online]. Available: <https://doi.org/10.48550/arxiv.2406.17249>.
- [72] P. Lajoie and G. Beltrame, "Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems", *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 475–482, 2024. DOI: 10.1109/lra.2023.3333742.
- [73] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

- [74] W. Chen, Y. Li, Z. Tian, and F. Zhang, "2d and 3d object detection algorithms from images: A survey", *Array*, vol. 19, p. 100305, 2023. DOI: 10.1016/j.array.2023.100305.
- [75] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
- [76] V. Kumar, N. Michael, and J. P. How, "Multi-robot coordination in warehousing", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 232–239.
- [77] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review", in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000.
- [78] L. E. Parker, "Alliance: An architecture for fault-tolerant multirobot cooperation", *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.
- [79] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018.
- [80] S. Thrun, "A probabilistic approach to concurrent mapping and localization for mobile robots", *Autonomous Robots*, vol. 5, no. 3, pp. 253–271, 2003.
- [81] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems", in *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, 1989.
- [82] B. Karin, A. Gonzalez, and X. Zhou, "A comparative analysis of stereo camera systems for depth sensing in robotics", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1234–1241.
- [83] S. Wang, L. Zhang, Z. Zhang, and S. Zhang, "Radar and camera fusion for robust object detection and tracking", *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 2035–2047, 2020. DOI: 10.1109/TITS.2019.2901681.
- [84] K. A. Tychola, I. Tsimperidis, and G. Papakostas, "On 3d reconstruction using rgbd cameras", *Digital*, vol. 2, no. 3, pp. 401–423, 2022. DOI: 10.3390/digital2030022.
- [85] Z. Ren and E. B. Sudderth, "3d object detection with latent support surfaces", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, Salt Lake City, UT, United States: IEEE Computer Society, 2018, pp. 104–112. DOI: 10.1109/CVPR.2018.00104.
- [86] N. V. K. Medathati, H. Neumann, G. S. Masson, and P. Kornprobst, "Bio-inspired computer vision: Towards a synergistic approach of artificial and biological vision", Inria Sophia Antipolis, Tech. Rep. 8698, 2016, fffhal-01131645v3f, p. 71.
- [87] R. Girshick, "Fast r-cnn", in *2015 IEEE International Conference on Computer Vision*, IEEE, 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.
- [88] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

- [89] P. Viola and M. Jones, "Robust real-time face detection", in *Proceedings of the Eighth IEEE International Conference on Computer Vision (ICCV 2001)*, Vancouver, BC, Canada: IEEE Computer Society, 2001, p. 747. DOI: 10.1109/ICCV.2001.937709.
- [90] Y. Wang, C. Wang, P. Long, Y. Gu, and W. Li, "Recent advances in 3d object detection based on rgb-d: A survey", *Displays*, vol. 70, p. 102077, 2021. DOI: 10.1016/j.displa.2021.102077.
- [91] Z. Zhang et al., "H3dnet: 3d object detection using hybrid geometric primitives", in *European Conference on Computer Vision*, Springer, 2020. DOI: 10.48550/arXiv.2006.05682.
- [92] C. Tao, H. Zhang, J. Liu, and J. Li, "F-pvnet: Frustum-level 3-d object detection on point–voxel feature representation for autonomous driving", *IEEE Internet of Things Journal*, vol. 10, no. 9, pp. 8031–8045, 2023. DOI: 10.1109/JIOT.2022.3231369.
- [93] J. Rodriguez, "A comparison of an rgb-d camera's performance and a stereo camera in relation to object recognition and spatial position determination", *ELCVIA Electronic Letters on Computer Vision and Image Analysis*, vol. 20, no. 1, p. 16, 2021. DOI: 10.5565/rev/elcvia.1238.
- [94] A. Paigwar, D. Sierra-Gonzalez, Ö. Erkent, and C. Laugier, "Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar", in *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, IEEE, 2021, pp. 2926–2933. DOI: 10.1109/ICCVW54120.2021.00327.
- [95] X. Zhou and G. Lin, "Review of target detection algorithms", *Frontiers in Computing and Intelligent Systems*, vol. 4, no. 3, pp. 17–19, 2023. DOI: 10.54097/fcis.v4i3.10736.
- [96] E. Arican and T. Aydin, "Object detection with rgb-d data using depth oriented gradients", in *Proceedings of the International Conference on Engineering and Natural Sciences (ICENS)*, Hungary - Budapest, 2017.
- [97] B. Karbouj, "Comparative performance evaluation of one-stage and two-stage object detectors for screw head detection and classification in disassembly processes", *Procedia CIRP*, vol. 122, 2024, Conference: 31st CIRP Conference on Life Cycle Engineering (LCE 2024), Turin. License: CC BY-NC-ND 4.0. Lab: Bsher Karbouj's Lab. DOI: 10.1016/j.procir.2024.01.077.
- [98] T. Eppenberger, G. Cesari, M. Dymczyk, and R. Dube, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking", in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. DOI: 10.1109/IROS45743.2020.9340699.
- [99] Z. Yin, H. Sun, N. Liu, H. Zhou, and J. Shen, "Decoratingfusion: A lidar-camera fusion network with the combination of point-level and feature-level fusion", *arXiv preprint arXiv:2501.00220*, 2024.
- [100] Y. Li, X. Ma, Z. Zhang, W. Ouyang, and Y. Zhang, "Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17182–17191.

- [101] Y. Li, Z. Zhang, X. Ma, W. Ouyang, and Y. Zhang, “Rethinking the late fusion of lidar-camera based 3d object detection”, *IEEE Transactions on Intelligent Transportation Systems*, 2024.