Progress Report I

Collective Transport using Decentralized 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 Srikhirin
6438187721 Mehul Sharma

2/2025: 2147417 Final Project II
Robotics and Artificial Intelligence Engineering (International Program)
International School of Engineering (ISE) Faculty of Engineering, Chulalongkorn University

Contents

1 Introduction **1**

2 SLAM **4**

3 Communication in the Swarm **6**

4 Odometry in the mobile robot **8**

5 Coordination in the Swarm **10**

6 Object Detection **12**

7 Conclusion **14**

Chapter 1

Introduction

This progress report aims to highlight the progress of our final project, **Collective Transport using Swarm Robotics**, with the evaluating criteria being the team individual contributions, as well as our pace in comparison to the ideal schedule. The ideal schedule can be represented by the project Gantt Chart (Figure 1.1).

According to Figure 1.1, we are exploring four major sub-tasks during this iteration of the project schedule. These four tasks are: **Communication in the Swarm** (Task 1.1), **Simple Simultaneous Localization and Mapping (SLAM) in Simulation** (Task 1.2), **Coordinated Gripping and Formation** (Task 1.3), and **Object detection** (Task 1.4). These tasks are planned to span from January to the second week of March. The results of the endeavors will be discussed in the following chapters

1. Preparation for Hardware Implementation
    - 1.1 Communication in the swarm
    - 1.2 Simple Simultaneous Localization and Mapping (SLAM) in Simulation
    - 1.3 Coordinated Gripping and Formation
    - 1.4 Object detection
2. Moving Towards a Complete Swarm
    - 2.1 Hardware
    - 2.2 Movement after Gripping
    - 2.3 Testing and Evaluation

To aid the visualization of this project. We have created a Petri Nets model (Figure 1.2) to represent the project's workflow. Petri Nets serve as a formal modeling tool to represent concurrent and distributed systems, providing a structured framework for analyzing complex workflows. In the transition(rectangle) "Task Assigned from Taskmaster" force synchronization between the two dependent ask "Wait for instruction" and "Planned Path for All" to start following the planned path. For the place "Reach each destination of Yellow Cylinder & grasp the cylinder", we can see it's

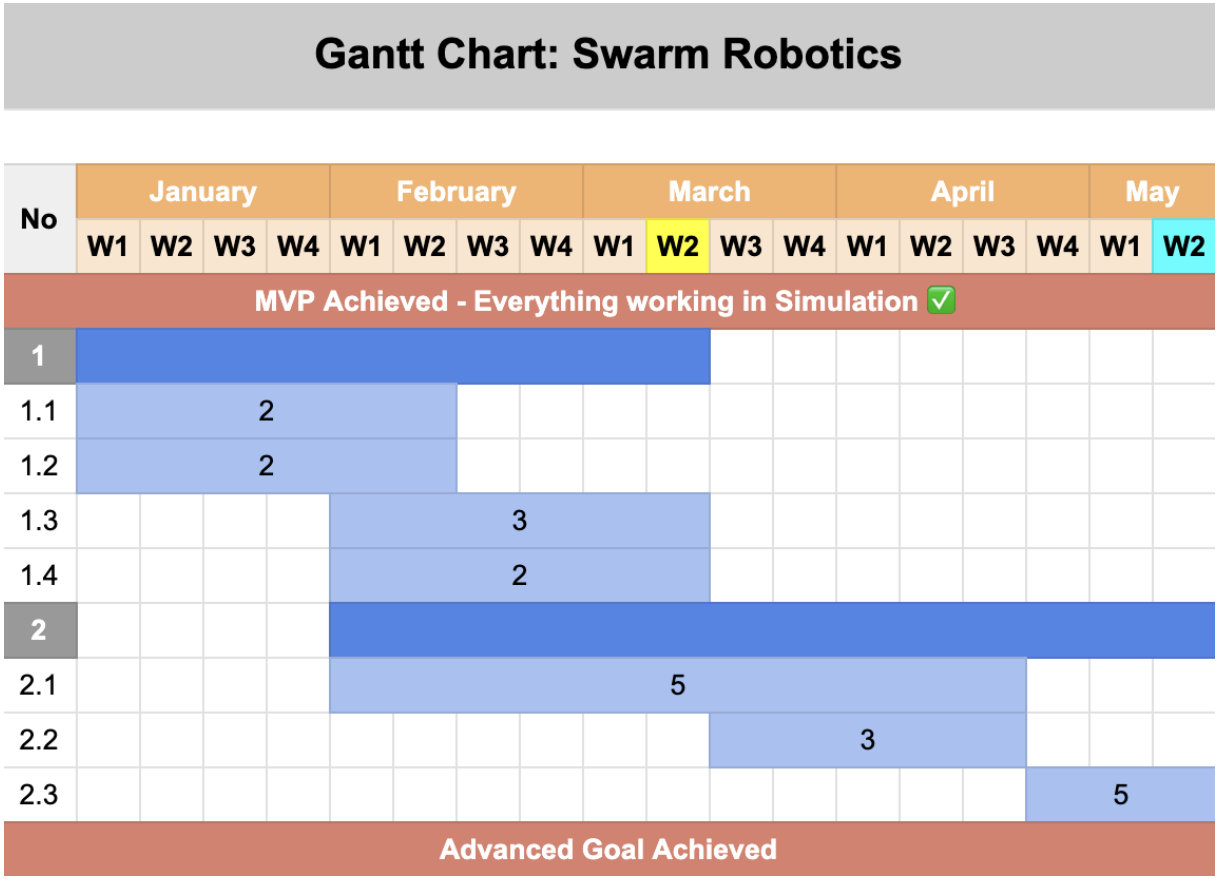

Figure 1.1: Project Gantt Chart

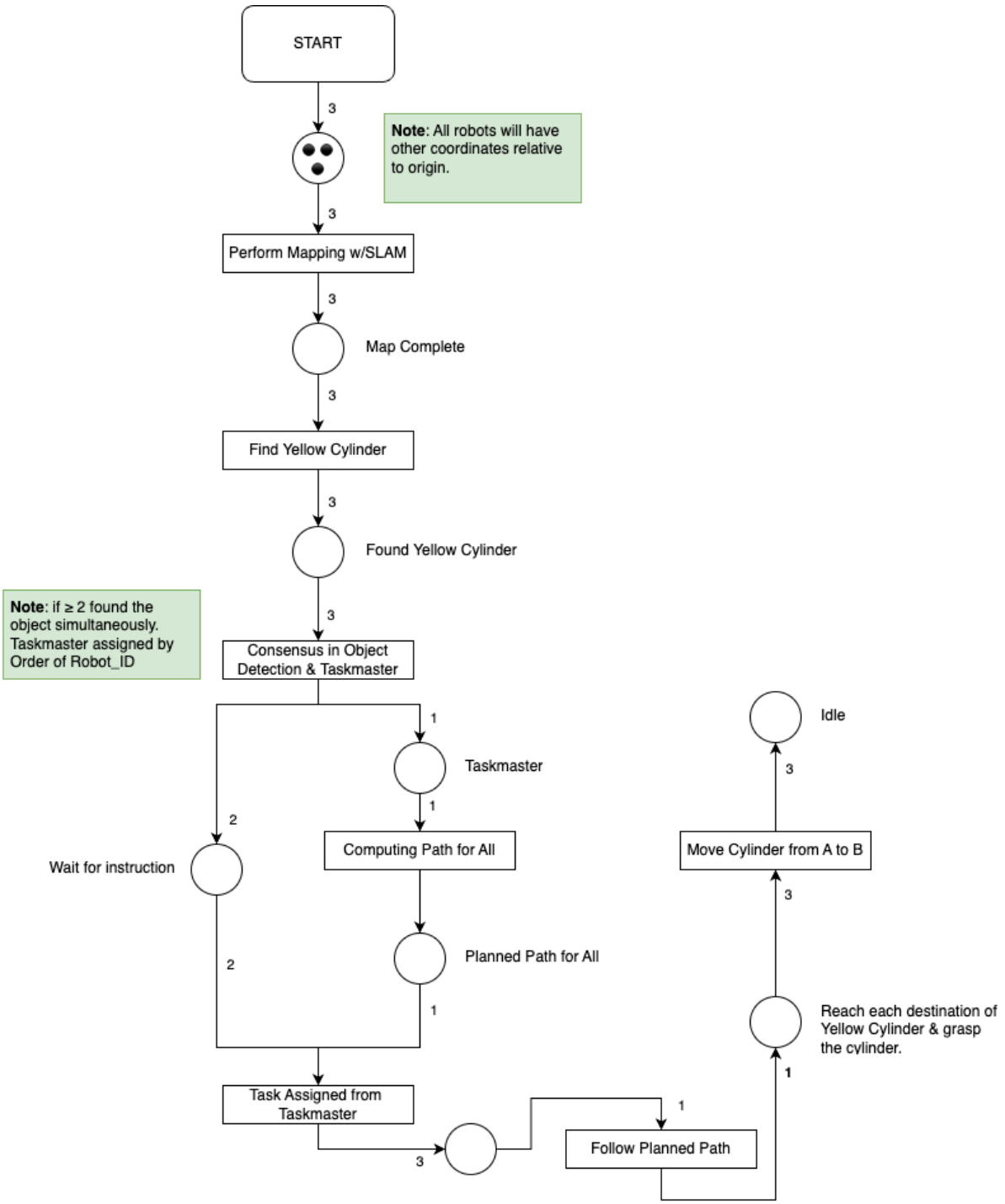

Figure 1.2: Petri Nets Model

Chapter 2

SLAM

In the previous semester, we selected GRAPH SLAM as our preferred SLAM approach. Initially, we planned to implement GRAPH SLAM within Webots using a GitHub package by Hobby Singh called Graph-SLAM. We specifically utilized the 2D LiDAR module, as our project solely relies on a 2D LiDAR without any camera input. However, we encountered challenges with this implementation because the package required a dataset composed of .clf files containing odometry and LiDAR logs. Although we successfully extracted the necessary logs, we continuously encountered an ICP error, suggesting that our dataset might have been missing certain fields or that the SLAM package needed fine-tuning for our specific project requirements. While the generated map accurately reflected the odometry poses and edges, the LiDAR readings resulted in unclear, hazy lines. After multiple attempts to resolve these issues, we decided to develop our own version of Graph SLAM while temporarily switching to the ROS2 SLAM package known as SLAM Toolbox.

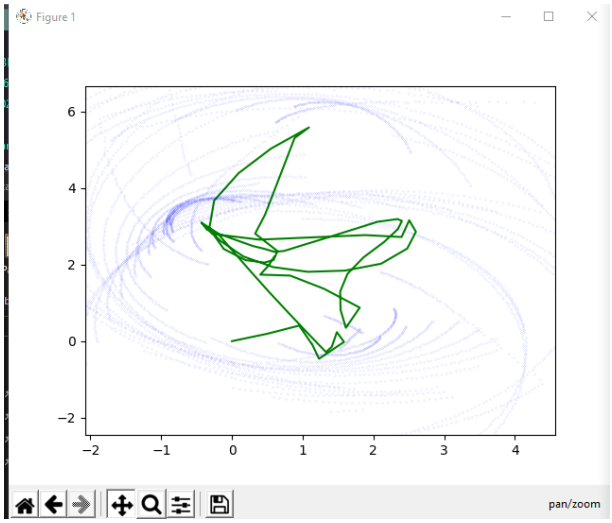

Figure 2.1: Map generated from the GRAPH SLAM package

SLAM Toolbox is essentially a graph-based SLAM solution that utilizes pose graph optimization to refine map accuracy and correct errors in localization. It represents robot poses as nodes and constraints (such as odometry and scan matching) as edges,

forming a graph structure that can be optimized to minimize drift and inconsistencies. This graph-based approach allows for efficient loop closure detection, ensuring that previously visited areas are correctly aligned, even in large or dynamic environments. By leveraging modern optimization techniques like Ceres Solver, SLAM Toolbox continuously refines the pose graph, making it a powerful and flexible tool for real-time mapping and localization in ROS2-based robotic applications. We initially tested SLAM Toolbox in a Gazebo simulation, where we successfully generated an accurate map. However, since this was purely in a simulated environment, transitioning to real-world implementation requires actual odometry data for accurate mapping and localization.

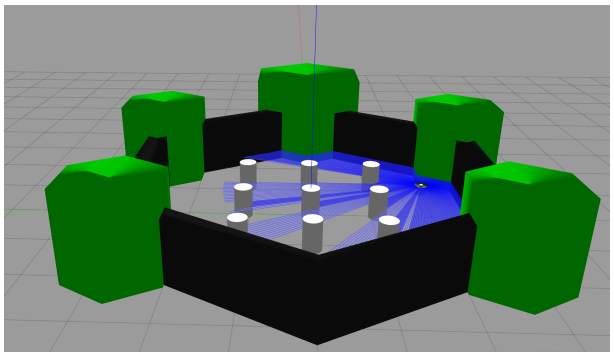

Figure 2.2: The Gazebo world

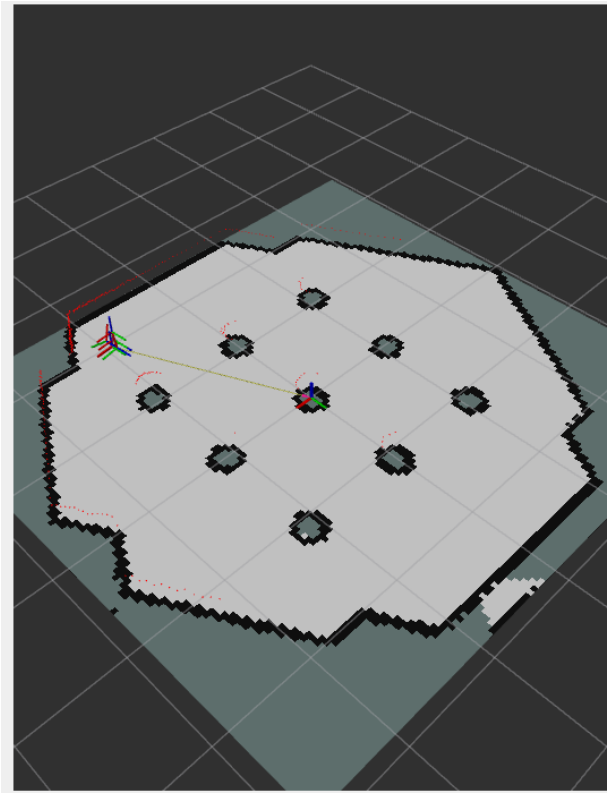

Figure 2.3: Map generated from SLAM Toolbox

Chapter 3

Communication in the Swarm

The Communication in the Swarm is eventually decided to be done in a centralized manner, by having a central computing unit control. The current implementation is to use socket communication to manage the swarm and broadcast messages with a consensus algorithm in case any member in the swarm has a conflicting claim in the role of becoming the swarm taskmaster. This inherently demands a relatively complex design.

To conduct a centralized swarm communication, the central computing unit for our demo is our computer to maintain connections and manage incoming or outgoing messages, with servers being a suitable alternative scalability-wise. However, by moving from our initial approach of decentralized swarm communication, the reduced reliability in the introduction of a single point of failure has to be addressed. This can be mitigated by utilizing servers being hosted in the cloud with horizontal scalability.

Additionally, if a centralized communication robotic system is our goal, then MQ Telemetry Transport (MQTT) protocol would also prove to be a viable option due to its simplicity in scaling because of its publish-subscribe architecture, as well as its efficiency and reliability due to the Quality of Service it provides. The key advantage for socket communication over MQTT for our use case would be because of its low latency. This decision is vital because for an increasingly large swarm, maintaining a large array of socket connections will increase its complexity proportionately.

For a centralized communication in the swarm, a high level overview is provided below (Figure 3.1).

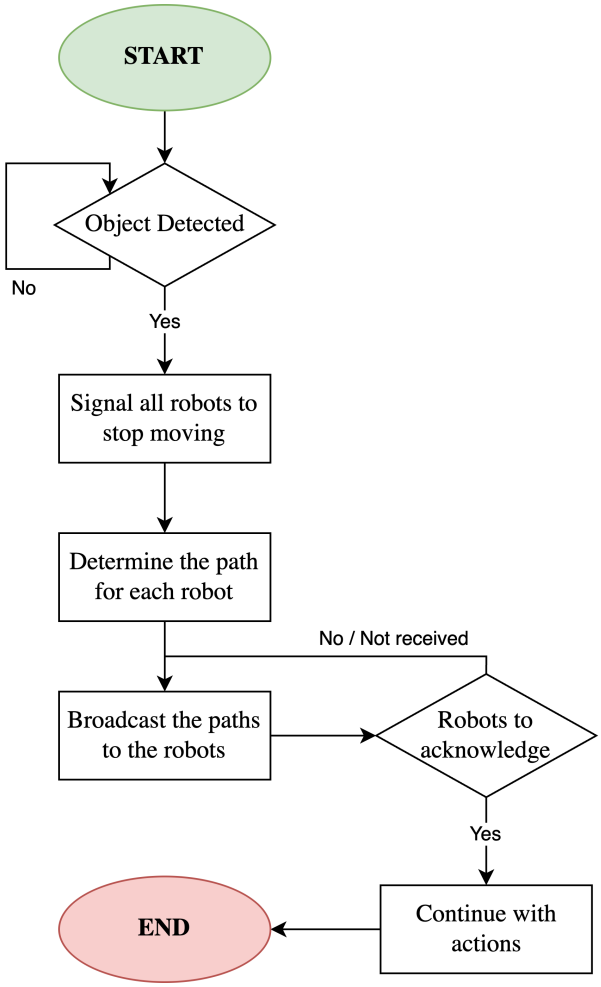

Figure 3.1: High-level Overview of the Communication Algorithm

Chapter 4

Odometry in the mobile robot

Currently, we have three main sources of information to predict a robot's position: wheel odometry, LiDAR odometry, and an inertial measurement unit (IMU). Each candidate has its own pros and cons in terms of accuracy and efficiency. This section will go through the tests and evaluation of each device and provide solutions to our problems regarding the robot's pose.

We first started investigating the IMU. The IMU we used was the AdaFruit BNO055 which is an IMU that has 9DOF. This is a versatile IMU Describe the advantage of Adafruit IMU here. We used an Arduino UNO board to communicate with the IMU. We were able to get the odometry from the IMU by using double Integration on the acceleration to get the x, y, and angular position. However, we noticed that the odometry was rift with inaccuracy. This is due mostly because of the integration drift. The inaccuracy means we cannot solely depend on the IMU for odometry.

The next odometry we investigated is the LIDAR odometry. For this we used a ros2 package called rf20-Laser. Every time the Lidar scans a point, a range flow constraint equation is formulated for each scanned point in terms of sensor velocity, and a robust function of the resulting geometric constraints is minimized to estimate motion. Unlike traditional methods, this approach does not rely on correspondences but instead performs dense scan alignment using scan gradients, similar to dense 3D visual odometry. After installing the ros2 package and running it, we found out that the odometry measurement is very accurate. However, the odometry is prone to drifting when the LiDAR is scanning a moving object which in our case will likely happen since we have three robots.

For wheel encoder odometry in four-wheeled omnidirectional mobile robots, the motor controls each wheel independently, resulting in total control of the robot's movements in the 2-D plane. The measurements from the rotational movements of the encoders can help us determine the distance traveled by the wheel through mathematical calculations. AX-12W presents a complete interface for reading and writing the motor status. As for wheel odometry, the main data we are interested in specifically are the present position and speed of the motor. With a baud rate of 1M, and the ability to bulk-read all motors instantaneously, we presume that our motors will be well adapted to the robot's pose estimation model. [1]

$$\begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\sin(\theta + \frac{\pi}{4}) & \cos(\theta + \frac{\pi}{4}) & R \\ -\sin(\theta + \frac{3\pi}{4}) & \cos(\theta + \frac{3\pi}{4}) & R \\ -\sin(\theta + \frac{5\pi}{4}) & \cos(\theta + \frac{5\pi}{4}) & R \\ -\sin(\theta + \frac{7\pi}{4}) & \cos(\theta + \frac{7\pi}{4}) & R \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \dot{\theta}_z \end{bmatrix}$$

Figure 4.1: Equation we use to calculate the wheel velocity, multiplication of the Inverse Jacobian matrix and the Twist message to get the wheel velocity. θ in this case is the orientation of the robot in the global frame. R is the radius of the robot from center to wheel. r is the radius of the wheel.

Sensor fusion is essential due to the inherent limitations of individual sensors. Wheel odometry is susceptible to wheel slippage and drift over time, LiDAR odometry performs well but is prone to errors when scanning dynamic objects, and IMU data suffers from integration drift. By fusing these sensors, their strengths can complement each other, mitigating individual weaknesses and improving overall accuracy. The Extended Kalman Filter (EKF) is the optimal solution for sensor fusion in this context because it effectively handles non-linear motion models while incorporating uncertainty from different sources. EKF recursively estimates the robot's state by predicting its position using a motion model and continuously updating it with sensor measurements, ensuring a more stable and accurate pose estimation. We could the feed this odometry to SLAM to achieve the best possible mapping.

Chapter 5

Coordination in the Swarm

This section talks about the lower level of robot coordination after the target object is found, and all robots surround it securely. Firstly, our robot can be controlled based on the dynamic model of an omnidirectional wheeled mobile robot. This allows our robots to obtain holonomic motion, the ability to move in the given x-y direction instantaneously without changing its pose. Hence, our team came up with a coordination method by taking advantage of holonomic constraints to plan the movements for each robot to move the target object.

Our model works on the assumption that all robots will not change their relative pose relative to the object, such that they move collectively like one solid object where the center is the object's centroid. For simplicity in the calculation, we assume that all robot is equally distributed around the object (Figure 5.1).

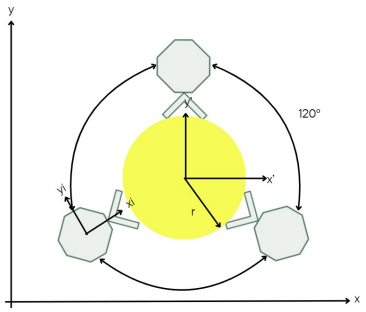

Figure 5.1: Robots and the object in global frame

Then, our main objective is to make the object traverse a given trajectory. For that, our robots translate the trajectory into their own using their positions and geometric constraints. Given that r is the distance from the robot's centroid to the object's centroid, and θ_i is the orientation of the i-th robot with respect to the object's frame, each robot adjusts its movement accordingly to maintain coordination in the system.

$$J_i = \begin{bmatrix} 1 & 0 & r\sin(\theta + \theta_i) \\ 0 & 1 & r\cos(\theta + \theta_i) \\ 0 & 0 & 1 \end{bmatrix}$$

Each robot's position is determined using a Jacobian transformation that accounts for its relative displacement from the central reference point. This transformation con-

siders the robot's angular offset and the radius of formation, ensuring that each robot maintains its relative position within the structure while responding to the overall system's movement. The Jacobian matrices allow the calculation of linear and angular velocities for each robot, ensuring accurate position updates over time.

These estimated velocities are included into the motion update equations to iteratively modify the orientation and position of every robot. The simulation represents the ongoing change of the motion of the system through discrete time increments. This makes it possible to observe the coordinated movement in which the robots dynamically modify their paths while maintaining the integrity of the formation.

These are results in MATLAB simulation:

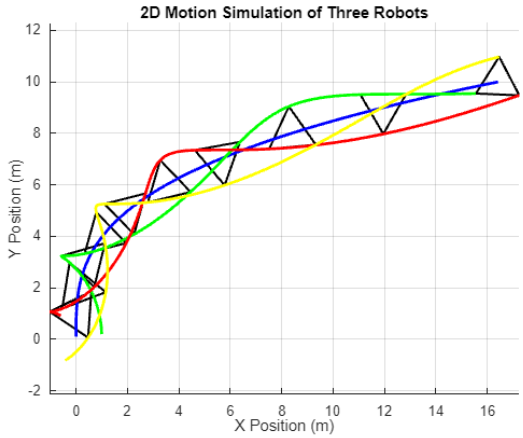

Figure 5.2: Parabolic function trajectory

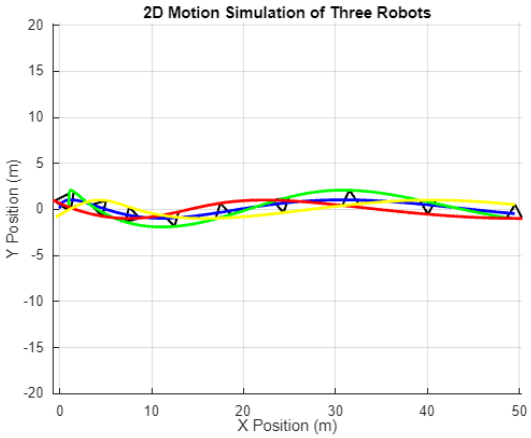

Figure 5.3: Sine function trajectory

Chapter 6

Object Detection

The object detection system initially developed in the previous semester successfully detected a yellow cylinder, generated a bounding box around it, aligned the box's center with the camera frame, and measured the distance between the robot and the object using the camera's focal length. However, this implementation relied on the assumption that the cylinder's dimensions were known, and LiDAR had not yet been integrated. Additionally, the scope of object detection was refined from hexagonal prisms and cubes to cylinders to eliminate the need for pose estimation and edge detection, which would otherwise introduce additional computational complexity for dynamic object grasping.

In transitioning to real-world implementation, several modifications were made to the object detection workflow. The system now identifies three distinct objects: a yellow, a blue, and a red cylinder, each differing in size. At this stage, only a single object is placed in the arena at a time, with detection based primarily on colour. Once detected, the system generates a bounding box around the object, and its distance and size are measured using an S3 RPLiDAR. To enhance accuracy, multi-sensor fusion between the camera and LiDAR is employed. By leveraging the camera's horizontal field of view (FOV), the system maps the bounding box's pixel range to the corresponding angle and retrieves the appropriate LiDAR data based on the robot's odometry.

The current implementation has successfully demonstrated the ability to detect objects based on color while filtering out reflections from the floor. Initial testing was conducted under controlled lighting conditions, including illumination from the left, right, front, and back. The presence of vibrations caused by the camera mount during movement did not significantly affect detection performance. However, at this stage, accuracy assessments are still based on visual inspection, and further validation will be conducted once the model undergoes proper calibration with the sensors. Overall, progress remains on schedule and aligns with the expected timeline, with each milestone being achieved as planned.

In measuring distance and size, image distortion remains a key factor influencing accuracy, as the system must correctly map the detected object's bounding box angle to real-world coordinates. Since the object's height does not contribute to measurement

calculations, vertical calibration data from the camera has been omitted. Additionally, the top 30 degrees of the camera frame is cropped to remove potential noise and prevent false detections of unintended objects or individuals outside the arena.

One of the primary challenges encountered in this phase is object occlusion, where multiple objects appearing in the frame can lead to overlapping bounding boxes. In such cases, the robot is required to reposition itself to separate the detected objects and ensure accurate size measurements without interference. Another challenge is ensuring reliable color detection under varying lighting conditions, as different light sources and angles can affect how the camera perceives object colors. Additionally, image distortion presents difficulties in mapping the bounding box to the correct position in relation to the LiDAR data. These obstacles are being addressed through careful calibration, adjustments in object positioning, and improvements to the detection algorithm to enhance its robustness against occlusion and lighting variations.

To meet the minimum viable product (MVP) requirements, the initial implementation is limited to detecting a single object within the arena. The system must be capable of generating a bounding box, measuring both distance and angle, and determining the object's size using LiDAR. Once these fundamental objectives are achieved, the next phase of testing will involve detecting multiple objects within a single frame while handling occlusion through coordinated robot movement.

Further testing will focus on optimizing the model and evaluating its accuracy under different conditions. The model will first be validated under the MVP scenario, dealing with only one object, before advancing to a multi-object setting with occlusion management. To assess distance measurement accuracy, objects will be placed at 0.5, 1, and 3 meters from the robot, allowing evaluation of the alignment between LiDAR and camera data. Additionally, eight different lighting conditions will be introduced by positioning the light source at varying angles to examine the robustness of the color detection model under diverse illumination environments.

Chapter 7

Conclusion

The progress for implementing effective communication in the swarm is behind schedule due to the move from decentralized to centralized with respect to communication. This allows for more flexibility in the protocol we can use for communication, with socket communication and MQTT being two major contenders. Once the advantages and disadvantages have been thoroughly considered, communication implementation will be able to continue with the most viable protocol for our use case as the solution.

Additionally, the object detection component remains on schedule, as the key aspects, including the detection model based on HSV and the extraction of distance measurements from the LiDAR, have been successfully completed. However, calibration and object occlusion handling will be implemented and tested once the necessary hardware is fully prepared for experimentation.

For the next phase of our project, we will complete the tasks that have yet to be completed during the previous iteration and move towards a complete swarm. This includes completing the hardware requirements, considering movement after gripping, as well as testing and evaluation. Additionally, we will proceed with other tasks as scheduled in the Gantt Chart to ensure all project milestones are met.

Bibliography

[1] A. Phunopas and S. Inoue, "Motion improvement of four-wheeled omnidirectional mobile robots for indoor terrain", *Journal of Robotics Networking and Artificial Life*, 2018.