December 2018

Coordination and Control for a Team of Mobile Robots in an Unknown Dynamic Environment

Sucheta Roy
University of Wisconsin-Milwaukee

COORDINATION AND CONTROL FOR A TEAM OF MOBILE ROBOTS IN AN UNKNOWN DYNAMIC ENVIRONMENT

by

Sucheta Roy

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Sciences

in Engineering

at

The University of Wisconsin-Milwaukee

December 2018

ABSTRACT

COORDINATION AND CONTROL OF A TEAM OF MOBILE ROBOTS IN AN UNKNOWN DYNAMIC ENVIRONMENT

by

Sucheta Roy

The University of Wisconsin-Milwaukee, 2018

Under the Supervision of Dr. Mohammed H. Rahman

This research presents a dual-level control structure for controlling a mobile robot or a group of robots to navigate through a dynamic environment (such as an object is moving in the workspace of a robot). The higher-level controller operates in cooperation with robot's state estimation and mapping algorithm, Extended Kalman Filter – Simultaneous Localization and Mapping (EKF-SLAM), and the lower-level controller (PID) controls the motion of the robot when it, encounters an obstacle, i.e., it reorients the robot to a predefined rebound angle and move it straight to maneuver around the obstacle until the robot is out of the obstacle range. The higher-level controller jumps in as soon as the robot is out of the obstacle range and moves the robot to the goal. The obstacle avoidance technique involves a novel approach to calculate the rebound angle. Further, the research implements the aforementioned technique to a Leader-Follower formation. Simulation and Experimental results have verified the effectiveness of the proposed control law.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Program Interface
ARIA	Advanced Robotic Interface for Applications
CVM	Curvature Velocity Method
EKF	Extended Kalman Filter
GPU	Graphics processing unit
HCR	Home Care Robot
IR	Infrared
LIDAR	Light Detection and Ranging
MST	Minimum-distance Spanning Tree
PID	Proportional Integral Derivative
SLAM	Simultaneous Localization and Mapping
VFH	Vector Field Histogram
WMR	Wheeled Mobile Robots

INTRODUCTION

Mobile robots are widely used today in a variety of fields such as agriculture, industry, land mining, military applications, space exploration, nuclear plants, and in many other applications where the environment is inaccessible or hazardous to humans. WMR have a greater degree of autonomy, which make them ideal for Lean operations, where robots help with increasing the efficiency of an industry or an organization. Most WMRs are equipped with an array of complex sensors to detect objects around them. The ability to accurately perceive a dynamic environment in real-time is incredibly difficult, but that is why WMRs are so valuable in a constantly shifting industrial setting. In the case of WMR, the problems of path planning, trajectory following, and obstacle avoidance are the most challenging and interesting subjects and would be addressed throughout the project.

To address this issue, as a part of this research we propose to develop an optimal path planning technique to make good use of any available information on the environment and to develop a control technique to drive the robot along the planned path. Also, the intent was to have a seamless communication of multiple robots, on the field, to a primary computer so that the location of the obstacles and the robots are recorded simultaneously to the common map. If no information is available about the environment or if a dynamic obstacle suddenly emerges, an obstacle avoidance technique that can autonomously drive the robots safely to the destination without colliding with objects on its way is desirable.

The research uses EKF-SLAM [12] to navigate around the room and to implement an obstacle avoidance method. This process was implemented by using a LIDAR and six IR sensors to scan

the immediate environment. The robot uses the scanned data to learn about its surroundings and to build a map of the previously unknown environment. The EKF-SLAM process continuously updates its location based off landmarks. Having created a map of the environment, the robot can use it to determine the best route for navigation. Another function of this research was to utilize the obstacle information to calculate and predict the optimum path to avoid the obstacle would be determined. The design objective of the research was to create a small and compact system which can be mounted on the robot without any interference with the navigation or other computer components. Detail description of the mechanical components are given in Chapter 3.

The goal of this research is to develop control and learning algorithms that will enables a group of robots to cooperate and work within a dynamic and in a completely unknown environment. The methodology would involve a combination of EKF and SLAM. The research would be validated on MATLAB using a group of three mobile robots to reach some pre-determined locations and return to the base or the origin. The hardware implementation of the algorithm on two mobile robots in the Leader-Follower formation would further corroborate the research.

The specific aims of this research project, based on the limitations outlined above, are:

- To develop the kinematic model of a three wheeled omni-directional mobile robot,
- To develop a group architecture which initiates coordination between the mobile robots,
- To construct some simple system dynamics where a generalized model is designed for the group of robots, and
- To implement EKF-SLAM to a team of mobile robots.

This project is divided into five phases, as follows:

Phase I – Design

In the design phase, all the necessary components that would be required, in order for the robot to avoid obstacles and communicate with the processing GPU, will be determined. The primary goal for the robot will be decided and its maximum capabilities were established. The chassis for the robot would be chosen so that the robots have a small size and have frames to mount sensors.

Phase II – Build and Program

In this phase, all the components will be mounted to the robot chassis. The initial programing to test all the motors will be completed, along with the calibration all the sensors. Further, initial tests will be done on the sensors and motors. Necessary changes would be made on the robot accordingly.

Phase III – Test

After the robot is assembled, it will be tested by a completely wired system. The robot would be tested extensively to verify that the robot detects an obstacle and consequently move through the predetermined locations. Troubleshooting and tweaking of the robot will be done in Phase IV to make sure the robot would perform as intended and that no unexpected errors arise once completed.

Phase IV – Adding Multiple robots

In this phase, multiple robots (one in our case) will be built in the same way as the first robot. The robot would be tested to effectively communicate with the processing GPU to simultaneously determine the locations of the other robots in a common map.

Phase V – Experimentation

After establishing communication between the robots and the processing GPU, the algorithm will be tested by simulating random static obstacles. After the simulation worked, the obstacle will be modified to be more complex, for this research, a circular wall with one opening would be designed. The next step would be to implement the hardware with the final algorithm and simulate real-time obstacles. Two tests will be conducted with the hardware. The first test directed the robots to predefined goals while avoiding static and dynamic obstacles. The next test instructed the robots to move randomly while avoiding all the obstacles. This phase would, therefore, conclude the objective of this project.

CHAPTER 1

LITERATURE REVIEW

Nowadays, groups of mobile robots are widely used in a variety of fields such as logistics robots and autonomous transport systems at plants, exploration, and navigation in space and nuclear plants, where the environment is inaccessible or hazardous to humans and even for educational purposes like use of Lego Mindstorm EV3 for FIRST Robotics challenges. The development of concepts for navigation for a group of robots is often a common research topic and has been studied extensively for finding an efficient strategy to avoid obstacles and for path optimization. Navigation in a dynamic environment presents a challenge for mobilizing and path planning for multiple robots. Although there are numerous solutions presented, there are plenty of hinderance associated with it, like for Heuristics algorithms [1], which can only be implemented for just static obstacles, or for MST [2], which needs to be re-computed at every time step and overloads communication, or in case of Gaussian Process Frontier-based Exploration [3], where dense coverage of obstacles leads to a noisy control. These are important issues since robots often must be able to exchange data and communicate with each other in order to coordinate and to successfully fulfill their tasks.

Baede, T.A., in [4] defines the system kinematics and implements a planar motion control algorithm of an omni-directional robot. The time-domain performance of the motion controller was tested under a variety of PID conditions using position control perfectly suited to the robot. The proposed strategies identified in the research removed the performance-degrading effect but could not accommodate velocity control due of its significantly lower performance in time-domain.

The motion of an omnidirectional robot with four wheels has been discussed in [5]. This research determined the problems in driving an omnidirectional robot, which are to detect and handle wheel slippage and to handle the failure of motors. Therefore, for an optimal control, the slippage in wheels are considered for the kinematics. The research also studied the effects of high velocity and accuracy of driving a robot in a real-life scenario using a PID control. Further, a control strategy is developed for the case in which the robot loses one of its motor.

A three wheeled omni-directional mobile robot, where the wheels are arranged at the vertices of an equilateral triangle, has been analyzed for its kinematics in [6]. In this research a very simple and easy to implement vector analysis on the wheels of the robot to determine its kinematic model is presented. Further, the wheel slippage was not considered for the analysis of the motion. This particular research uses a low-level program to achieve basic robot functionalities like turning around, moving forward, etc., which can easily be built up to a complex program where the robot can maneuver diagonally.

There are other algorithms such as Sequential Decision Theory, Feedback Motion Planning, various Differential Models which gives an optimum means to develop a trajectory and path planning and following system for a particular type of robot. For example, Maalouf in [7] adopted a technique to assess terrain and generate a cost matrix for a 3D terrain for the trajectory planning for the Pioneer 3-AT robot has been analyzed. It uses fuzzy controller with a terrain smoothness factor to follow the planned trajectory. Further, a fully autonomous navigation system was

implemented using the ARIA interface for localization and CVM for obstacle avoidance. The algorithm worked well for simple obstacle objects but was not debugged for complex obstacles.

Mojtahedzadeh, in [8], presents an obstacle avoidance model using the Kinect sensor. The algorithm presented is primarily developed in MATLAB and uses the straight-line segments and circle arcs methods along with the point cloud data extracted from the Kinect to model the surrounding environment. The data clustering process is used to group the point cloud data in such a way that it resembles the obstacles. This algorithm can be adapted for a LIDAR and/or IR sensor, which are used to develop various small-scale robots, as has been used for this research. But the research suffered due to the projection pattern of Kinect, which was not able to detect very bright and dark areas. Further, the limited range detection of Kinect limited the research to indoor environment.

The research presented in [9], provides a simple real-time obstacle avoidance algorithm for mobile robots. The proposed algorithm calculates a rebound angle based on the data collected equally spaced ultrasonic sensors. The primary advantages of the algorithms were that it has a very less computational load and can be used for low cost sensors and microcontrollers. Some of the drawbacks are that the sensors are highly subjected to noise and further, implements less data points collected from the surroundings. This algorithm, therefore, has the potential to be modified and be implemented to a higher-level programming with sensors like LIDAR which can collect multiple data points from all around the robot.

For a group of robots, one of the most primary concern is to control and update each of the robots' location in the map, so that the robots don't interfere with another robot's trajectory. The leader-follower approach is one of the most implemented method for formation control problems, as described in [10-11]. The research done in [10] defines a chain structure with two Leaders but due to lack of correcting spacing errors, the errors gets amplified considerably down the chain of robots. Approaches for the leader-follower formations, like non-smooth analysis with synchronization of navigation around artificial potential fields to maintain a triangular formation, has been discussed in [11]. This research based its navigation feedback based on the partial state of the leader received by the followers due to a time-varying velocity, which eventually leads to uncertainty when obstacles are taken into consideration.

A combination of the traits mentioned by the aforementioned researches can be useful for successfully having a formation control in a dynamic environment. Techniques like SLAM where a robot enters an unknown environment and collects surrounding information to spatially locate itself and other obstacles in a map can further contribute to an effective localization and mapping. In [12], SLAM has been used with an EKF estimator to reduce uncertainties in the system. In the proposed algorithm, the robot observes landmarks (obstacles) to locate itself or correct its location in a space. This technique can be extended to a more complex program to achieve robot mobility in real-time.

In case of a multiple number of robots in a system, the primary issue to look at is how to establish an effective localization strategy so that one robot does not interfere with another. Exploring an unknown environment by a group of robots has been discussed extensively in [13-15]. Instead of

building a powerful single robot, a WMR group can provide flexibility in performing the task required, as well as making the system more tolerant to possible individual robot faults. Most of the robots use a global map to plan their paths to efficiently coordinate their actions. Using the information collected from a robot about an explored cell and simultaneously updating the global map, the robots move to their respective targets.

In [13], a target point is assigned to a specific robot so as to minimize the time needed to completely explore an environment. The robots use a global map to plan their paths to efficiently coordinate their actions. The concept of grid maps has been used where each cell gets assigned a probability of having an obstacle. Using the information collected from a robot about an explored cell and simultaneously updating the global map, the robots move to their respective targets. The primary drawback of this research was that the robots were not able to distinguish between targets assigned to another robot.

The research discussed in [14], uses several strategies, referred as Reserves, Divide and Conquer, and Buddy System for multi-robot spatial exploration. In Divide and Conquer, the group of splits in two to accomplish a particular task, in Reserves, extra robots in waiting until they are needed for the task and in the Buddy System, the group of robots split in two to achieve two separate tasks. The Reserves strategy is significantly slower at exploring the environment than other strategies. But all the strategies had some amount of interference between the robots.

Mehrjerdi, in [15] proposes a dynamic tracking control using a two-level controller to deliver smooth robot movement. The lower level controller uses a simple PID whereas the higher-level

controller uses a feedback controller using an exponential sliding mode, a Lyapunov technique for coordination between robots and fuzzy control to instruct the robots to avoid any obstacles along their paths. The robot effectively avoids collision to another robot by simply moving out of its way, but the strategy presented cannot regain its path if it loses its coordination, after a collision.

Coordination techniques like static and dynamic and communication has been detailed in [16]. Further, two control architectures, centralized and decentralized, were elaborated. The advantages and disadvantages of both were discussed. A centralized control architecture proved to be better for a small group of robots in a relatively quiet environment. For the decentralized control architecture, a hierarchal and distributed control was defined. The decentralized control proved to be more reliable and robust, but the control is not as effective as that of the centralized control where a single computer can manage the path of all the robots.

In this research, a dual-level controlling algorithm has been discussed. A higher-level controller is developed based on SLAM with EKF as discussed in [12]. The lower-level controller is used when the robot encounters an obstacle. The higher-level controller activates as soon as the robot is out of the range of the obstacle/s. A novel obstacle avoidance technique has been described which makes the obstacle avoidance algorithm very easy to implement with different control algorithms. The Chapter 2 of the paper describes the setup of the system the proposed method has been tested upon. Chapter 3 and 4 goes over the Kinematic Modelling and the Obstacle Avoidance method, Control Approach and, the Mapping and Localization, respectively. Finally, the paper establishes the results derived by the simulation and hardware implementation (Chapter 5) and presents a conclusion (Chapter 6).

CHAPTER 2

KINEMATIC MODELING

This Section of the research goes over the Kinematics of the HCR with Omni wheels. Further, this chapter explores techniques enable an efficient, smooth and continuous robot movement along a desired path determined by the control algorithm discusses in Chapter 3. Moreover, the Formation Control Method and the Obstacle Avoidance method for a group of three mobile robots has been discussed later in this section.

2.1. Modelling the Motors

This section aims to describe the kinematic model for robot with omni directional wheels as shown in Figure 1. Therefore, via the suitable structure design, this kind of the mobile robot base can make the robot move any direction, and hence construct omnidirectional maneuvering capability.

Figure 1: Omni directional wheels

Figure 2 depicts the kinematic diagram that is used to find the kinematic model of the robot, where θ denotes the vehicle orientation. Velocity components in the x and y direction, V_x and V_y, can be determined by the linear velocity V and the rotating angle θ, as shown in Figure 2.

$$V^2 = V_x^{\,2} + V_y^{\,2} \qquad V = \sqrt{V_x^{\,2} + V_y^{\,2}}$$

$$V_x = V\cos(\theta) \qquad V_y = V\sin(\theta)$$

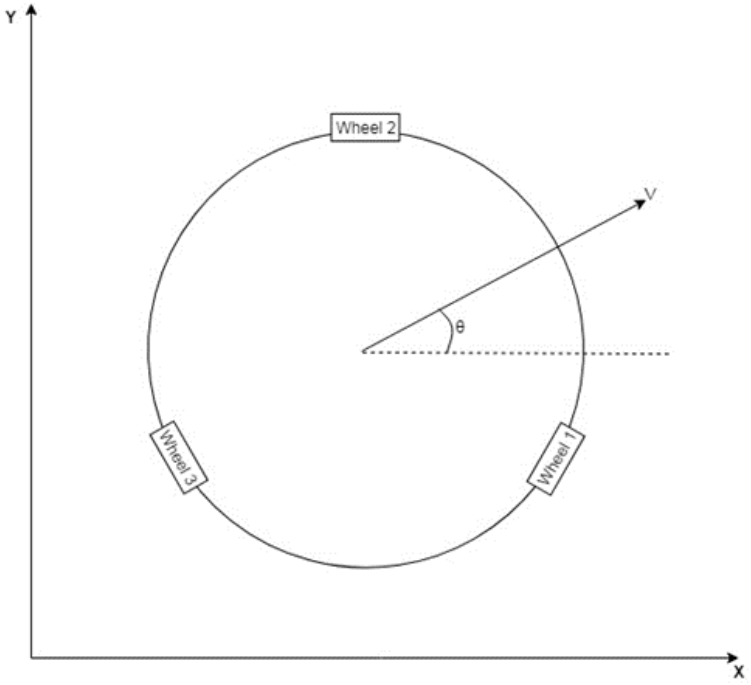


Figure 2: Rotation with respect to the robot center

The kinematic model of the mobile robot is based on the analysis of the rotation of the wheels. The velocity of each wheel is calculated using Vectors method. The vector is determined for each wheel, as shown in Figure 3, for the robot placed at the origin in its initial position.

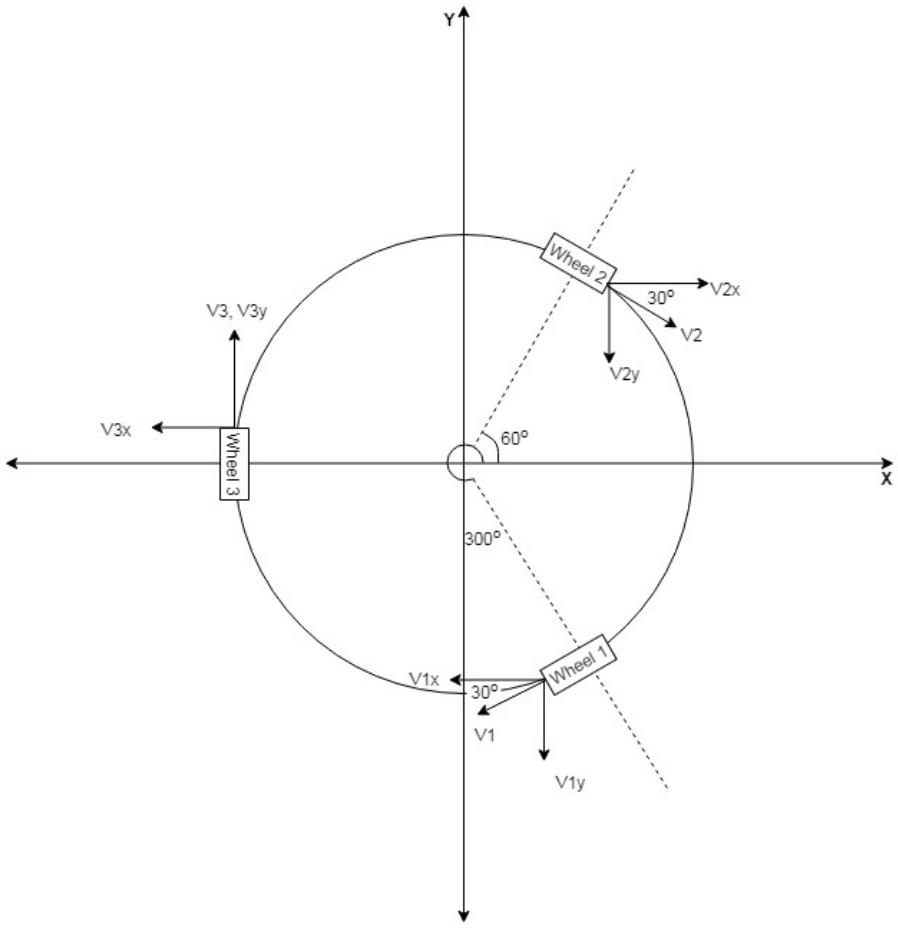


Figure 3: Wheel vector analysis for Omni-wheeled drive

The x and y component for each wheel with respect to the wheel velocities V_1, V_2 and V_3 can be written as,

$$V_{1x} = -V_1 \cos(30°) \qquad V_{1y} = -V_1 \sin(30°)$$

$$V_{2x} = V_2 \cos(30°) \qquad V_{2y} = -V_2 \sin(30°)$$

$$V_{3x} = V_3 \cos(90°) \qquad V_{3y} = V_3 \sin(90°)$$

Also, the angular velocity $\dot{\theta}$, for the robot can be written as,

$$\dot{\theta} = \frac{V}{L} = \frac{V_1 + V_2 + V_3}{r}$$

here, L is the radius of the robot base and r is the radius of the wheels.

$$V_x = -V_1\cos(30°) + V_2\cos(30°) + V_3\cos(90°)$$

$$V_y = -V_1\sin(30°) - V_2\sin(30°) + V_3\sin(90°)$$

By combining the three equations for $\dot{\theta}$, V_x and V_y, we get,

$$\begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -\cos(30°) & \cos(30°) & \cos(90°) \\ -\sin(30°) & -\sin(30°) & \sin(90°) \\ \frac{1}{r} & \frac{1}{r} & \frac{1}{r} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$

By inverting the coefficient matrix, we can get the wheel velocities as,

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} -0.5774 & -\frac{1}{3} & \frac{1}{3} \\ 0.5774 & -\frac{1}{3} & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

Thus, we can get the equations for the wheel velocities as,

$$V_1 = -0.5774V_x - \frac{1}{3}V_y + \frac{1}{3}\dot{\theta}$$

$$V_2 = 0.5774V_x - \frac{1}{3}V_y + \frac{1}{3}\dot{\theta}$$

$$V_3 = \frac{2}{3}V_y + \frac{1}{3}\dot{\theta}$$

2.2. Formation Control

This research defines a leader-follower approach with a triangular formation for three identical robots. The follower robots are equally spaced from the leader and the formation is shown by Figure 4.

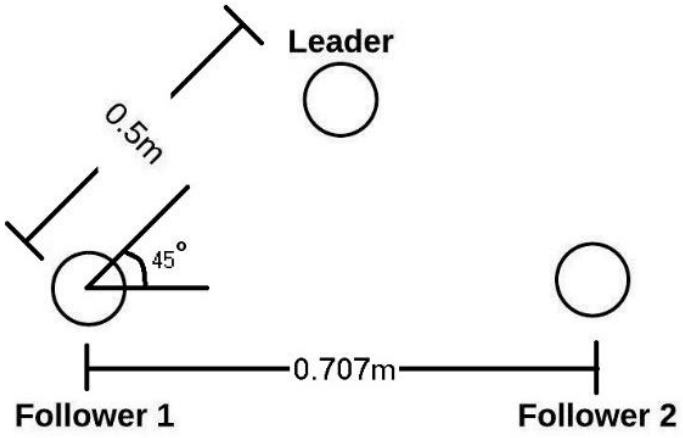


Figure 4: Leader-Follower Formation

The leader and the followers, both follow the same angle of rotation and are always facing the same direction. The position of the followers can be estimated by vector analysis as shown in Figure 5.

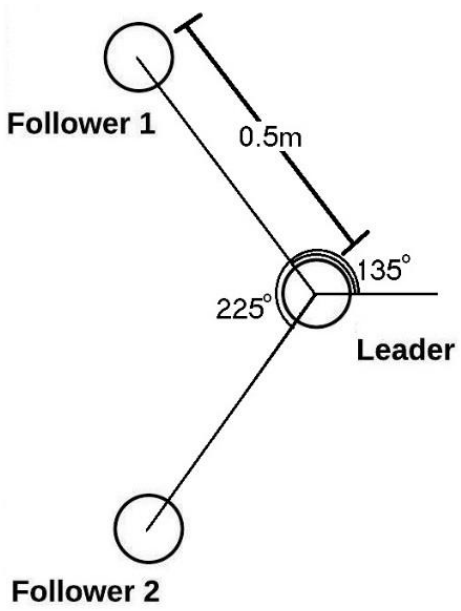


Figure 5: Followers position and orientation

For follower 1,

$$x_1 = x + 0.5\cos(\theta + 135)$$

$$y_1 = y + 0.5\sin(\theta + 135)$$

For follower 2,

$$x_2 = x + 0.5\cos(\theta + 225)$$

$$y_2 = y + 0.5\sin(\theta + 225)$$

2.3. Obstacle Avoidance

The obstacle avoidance method uses a novel modified version of the combination of the Bayes theorem, which states the probability of an event given the probability of a known event, along with the VFH method, where it calculates a rebound angle, which is the angle

at which the robot should rotate to avoid collision with an obstacle, for the robot by determining the lowest obstacle density.

For the proposed method, an efficiency quotient is calculated which determines the factor using the equation shown below and Figure 6, shows how the robot observes its surrounding.

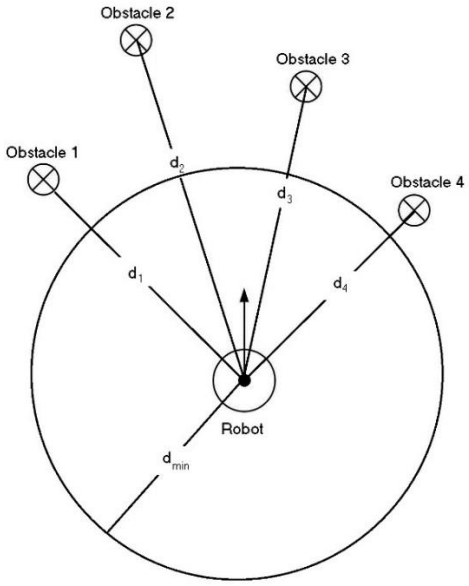


Figure 6: Robot Observation of Obstacles

For the purpose of this research, the obstacles are considered as circular objects with the radius equal to the radius of the robot, whereas, the robot is considered as a point in the 2D space.

The efficiency quotient is calculated by obtaining a ration of the distances decreases as the obstacle distance increases from the robot. The efficiency quotient's value is between 0 and 1, for an obstacle range of 1m to 2m, 0 being the robot should avoid the angle and 1

being the angle the robot can go in the particular direction without any interaction with any obstacles.

$$\eta_i = \left|1 - \frac{\mathrm{d}_i}{d_{min}}\right|$$

where, η = Efficiency quotient, d_i=Distance between the obstacle and robot center, d_{min} = Minimum distance at which the obstacle is detected

The formula given in the equation below uses the efficiency quotient. The numerator is the sum of the resultant of the product of the efficiency quotient and the slope of the obstacles detected by the robot. Since there are multiple sensor readings, an optimum rebound angle is determined using equation below,

$$\text{Rebound Angle} = \frac{\sqrt{(\sum_{i=1}^{r} slope_i \times \eta_i)^2 + (\sum_{i=2}^{r-1} \mathrm{slope_i} \times \eta_i)^2}}{\sum_{i=1}^{r} \eta_i}$$

where, r = Number of robots, slope = Angle of the obstacle w. r. t the robot pos

CHAPTER 3

TECHNICAL DESCRIPTION AND SYSTEM ARCHITECTURE

This chapter presents all the hardware components use for the successful completion of this research. Some of the key aspects for the robot is that it should have a small base frame so that it can move around small spaces along with multiple follower robots. Also, the sensors should have an unobstructed view of its surroundings. Furthermore, the system architecture, which provides supports for behaviors of the system hardware interface, has been discussed in this chapter.

3.1. Physical Components

The hardware of this research comprised of two fully functioning robots, 2 Microcontrollers, 2 pairs of XBee S1 modules with a pair of Shields and USB Adapters and 12 IR Sensors. Moreover, one Lidar was mounted on the Leader robot to collect additional obstacle data,

3.1.1. HCR Robot

The control algorithms developed in this research have been tested on the HCR robot. The HCR (Figure 7) is holonomic robot with 3 Swedish 90^{o} omni-wheels. The robot is equipped with 3 12V DC motors with a maximum speed of 122rpm each. Therefore, the maximum linear velocity of 1 m/sec, is calculated using vector algorithm, can provided to the robot algorithm. As a result of having omni-directional wheels, the robot has unique capability to move diagonally. Further, the robot has a span of diameter 31.5 cm, which is small and compact for small-scale research purposes.

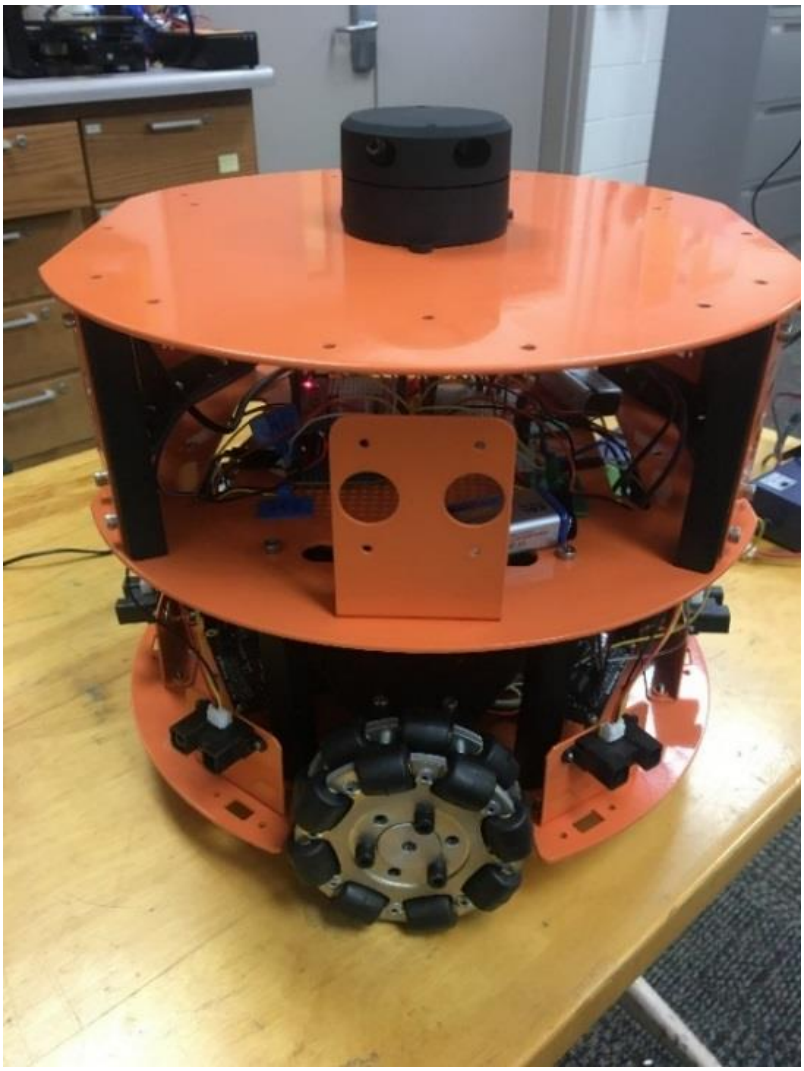


Figure 7: HCR Omni wheeled robot with LIDAR

3.1.2. Bluno Mega 2560 Microcontroller

The Bluno Mega 2560 (Figure 8) is very similar to Arduino Mega 2560 with an additional Bluetooth capability. This microcontroller is made by DF Robots, who also manufactured the HCR mobile platform. The microcontroller is Arduino software compatible and delivers the same performance as any other Arduino boards.

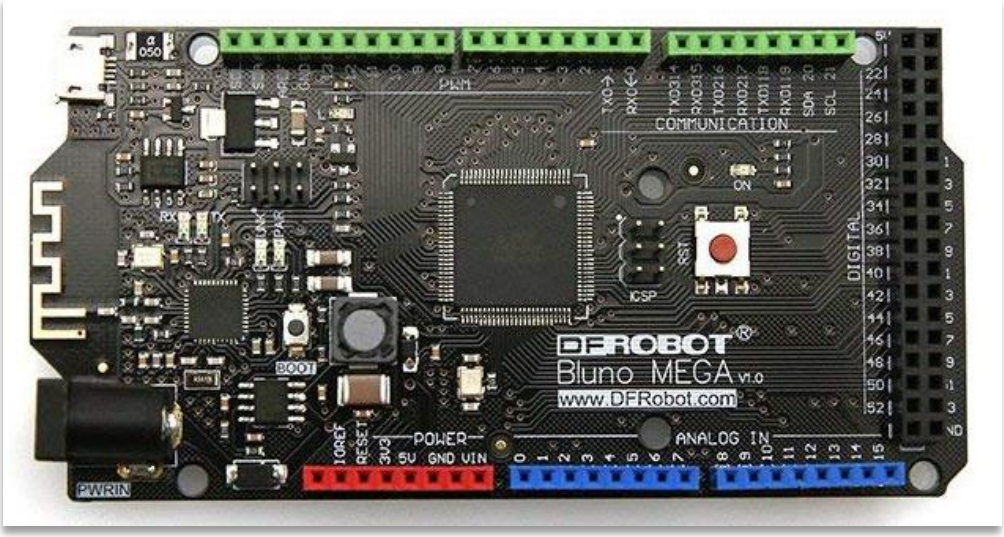


Figure 8: Bluno Mega 2560 Microcontroller

It has 54 digital I/O (input/output) ports, 15 of which can be used as PWM output, 16 analog input, and 4 UART. The power supply will be automatically switched. The range of voltage limit of the control board is from 5V to 23V.

3.1.3. Wireless Serial Communication

Two XBee S1 modules are configured to act as both a receiver and a transmitter. One Xbee module is mounted to the robot via a Sparkfun Arduino XBee Shield (XBee 1) and the other XBee module is attached to the main PC via an USB adapter (XBee 2) as shown in Figure 9. The main function of XBee 1 is to send the collected IR sensor data by the Arduino when XBee 2 sends a command to read the IR sensors. Whereas XBee 2 receives the IR data from XBee 1 and after the computation send the command and computed individual wheel velocities to XBee 1 to run the robot.

Figure 9: XBee S1 modules with Sparkfun Arduino Shield (Left) and USB Adapter (Right)

These modules are reliable and simple point to point and multi-point networks are easy to implement with Bluno Mega 2560 to get data and transfer it to a main computer via serial.

3.1.4. IR Sensors

The robot is equipped with 6 Sharp GP2Y0A02YK0F IR sensors (Figure 10). There are 3 pairs of sensors, each located between two wheels of the robot, and are mounted on the pre-built sensor frame of the robot. This device outputs the voltage corresponding to the detection distance. So, this sensor can also be used as a proximity sensor. The sensors have a measuring range of 20 to 150 cm.

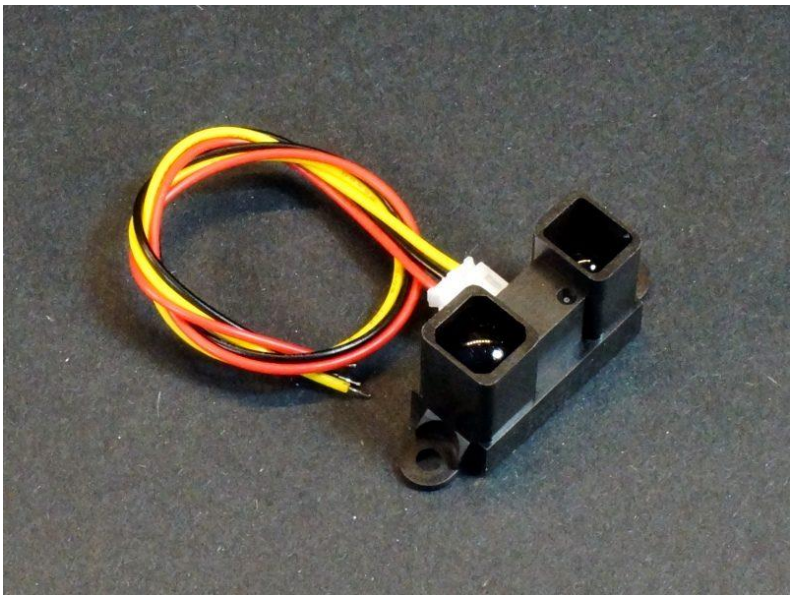


Figure 10: GP2Y0A02YK0F IR Sensor

The sensors are mounted on the robot as shown in Figure 11. The distance between a pair of sensors is 13 cm. The location and the angle at which the sensors are positioned are used for determining the location of the obstacles with respect to the center of the robot.

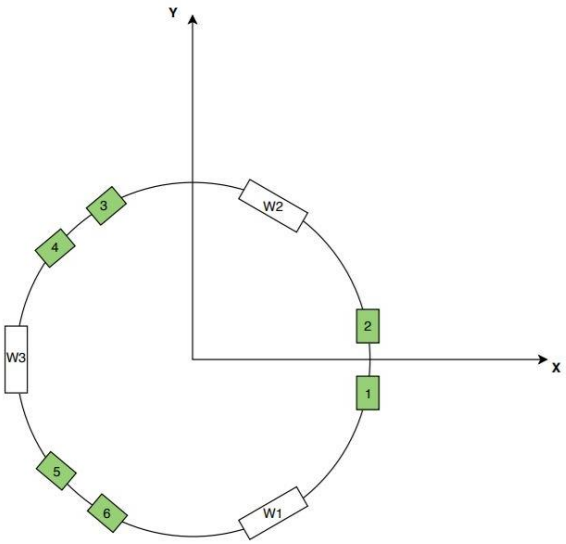


Figure 11: Schematic drawing for Sensor placement on the HCR robot

The IR sensors are read by the Arduino module as a voltage. Therefore, the voltage needs to be converted to a distance reading. Hence, by analyzing the analog voltage readings at every 5 cm from 25 cm to 145 cm (Appendix A) and curve fitting as shown in Figure 12 was done to obtain the equation relating the distance (d) of an obstacle from IR sensors to the analog voltage (v) read by the Arduino for the IR sensor.

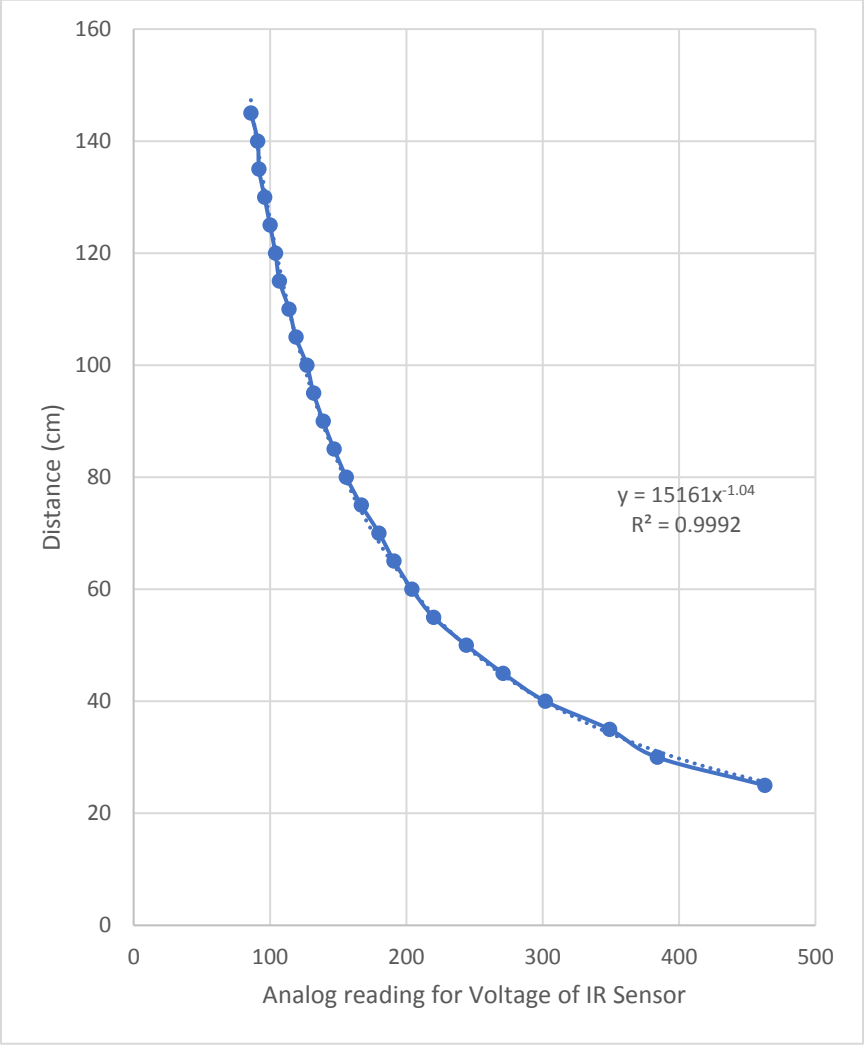


Figure 12: Analog Voltage readings to the Distance from the IR Sesnor

The following equation was determined to calculate the distance,

$$d = 15161 \times v^{-1.04} \quad (m)$$

The distance represented by the above equation is with respect to the sensor it is read out from. Therefore, each sensor reading is then converted using simple vector components to represent the distance from the center of the robot. These readings would be used for obstacle avoidance and, would be transformed to a global coordinate system to map and update the obstacles' and the robot's position and orientation.

3.1.5. Lidar

The YDLIDAR G4 Lidar (Figure 13) is a 360-degree two-dimensional distance measurement sensor. The Lidar is based on the principle of triangulation and is equipped with relevant optics, and algorithm design to realize high-frequency and high-precision distance measurement. At the same time as the distance measurement, a 360 degrees of scanning distance measurement is achieved by continuously obtaining the angle information through the 360-degree rotation of the motor. The Lidar was used as an additional sensor to observe obstacles in the surroundings. The Lidar is very effective in getting a range scan point cloud data through the Serial port.

Figure 13: YDLidar G4

For the purpose of this research, the Lidar uses ROS packages which are sourced by the manufacturer, to run and collect surrounding obstacles data. The Lidar communicates using a 3.3V level serial port (UART). The serial port is used to send data over to MATLAB for further computation. The polar coordinate of the system is based on the center of the rotating core of G4. The specified angle is clockwise positive. The zero angle is located at the exit of the G4 interface cable, as shown in the Figure 14.

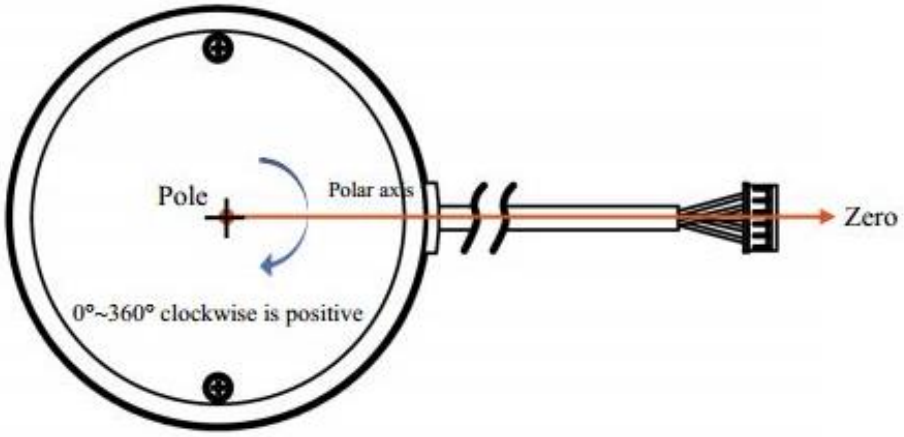


Figure 14: Lidar Configuration diagram

The data extracted from the LIDAR are found with respect to the orientation of the LIDAR on the HCR robot. The LIDAR is placed on the top of the robot and is centered with the robot's coordinates, so that the LIDAR reads the data with respect to the robot position and orientation.

3.2. System Architecture

3.2.1. Block Diagram

The block diagram shown in Figure 15 represents the current system with which the results have been generated. This system is completely a wired setup except the transmission of data via the two XBee modules.

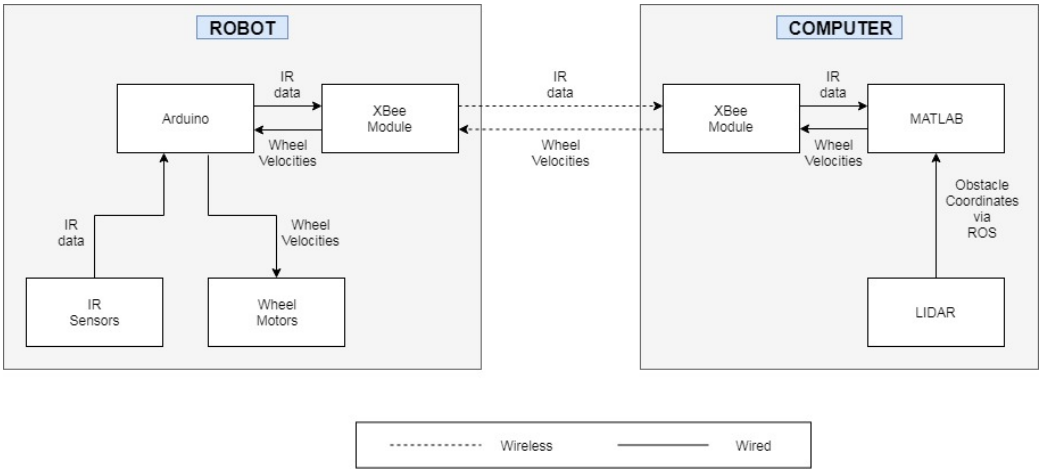


Figure 15: Block Diagram representing the System Architecture for the HCR

3.2.2. Experimental Setup

Figure 16 shows the complete system design of the experimental tests. A Zigbee technology communication device is used between the application program with 115200 baud rate, implemented in MATLAB, and the mobile robots. The mobile robots used in the experimental setup have three omnidirectional wheels actuated with DC-motor.

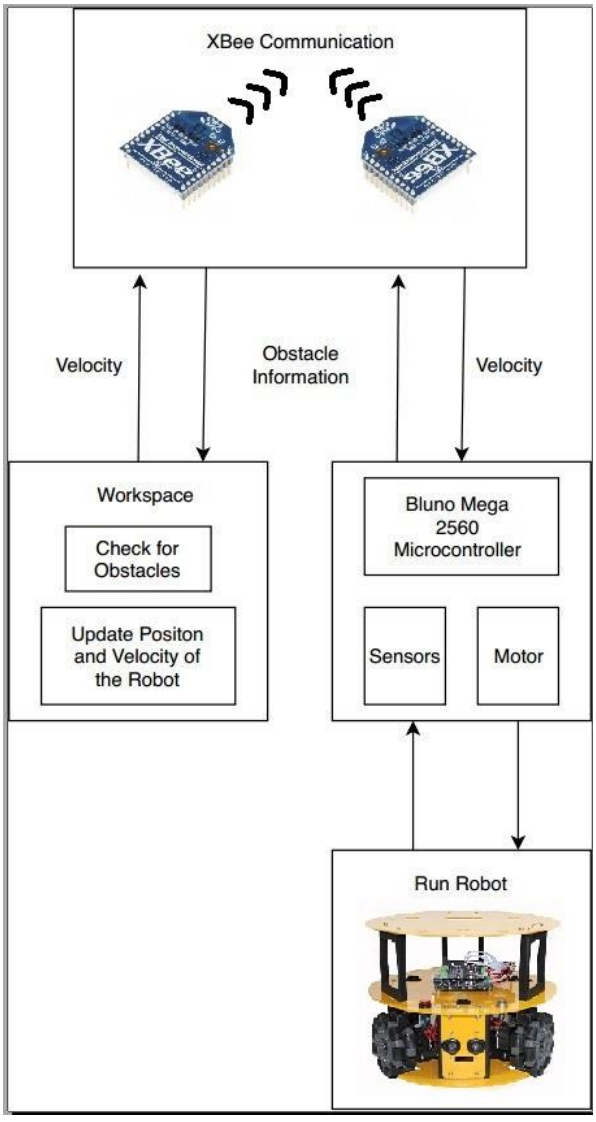


Figure 16: Hardware Setup

3.2.3. Communication Protocol

In the communication protocol, shown in Figure 17, the serial communication between the Robot and the GPU has been presented. The serial receives a command (1 or 2) from MATLAB and instructs the Arduino presented on the Robot to either read the IR Sensor or read the wheel velocities to run the motors.

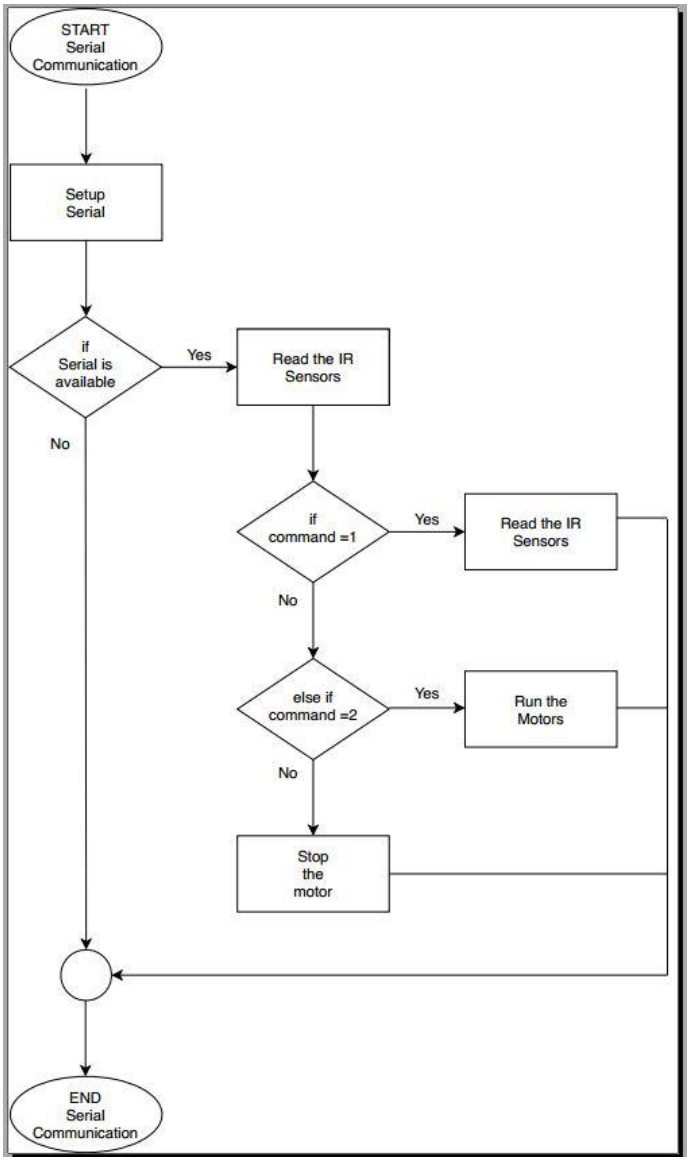


Figure 17: Communication Protocol

3.2.4. Controls Schematic

The complete system design of the control has been shown in Figure 18. The schematic defines the control process implemented by the algorithm on MATLAB. The process starts with receiving the data from the Sensors on the Robots and then checking for any obstacle detected. The Update section involves running the higher and lower level controller, discussed in Section 4.3 and 4.4. The Kinematics described in Chapter 2 are then implemented to the Robots.

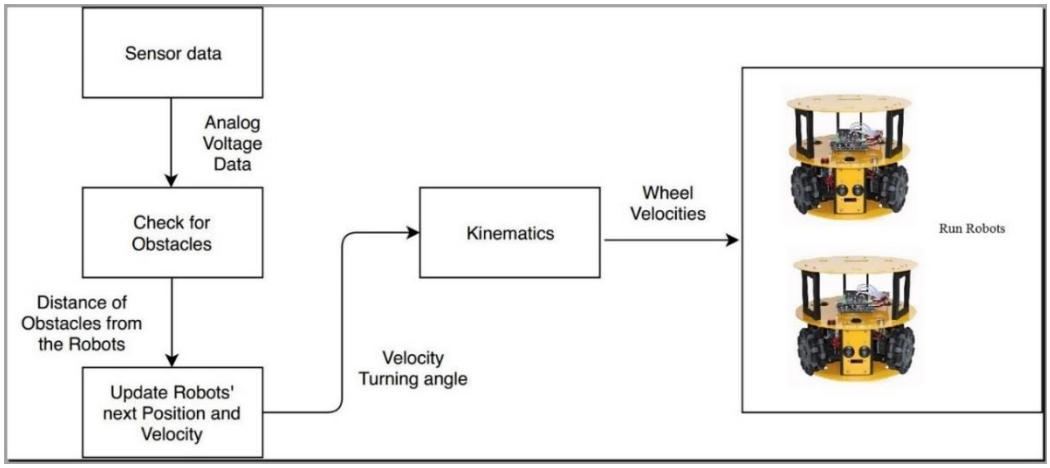


Figure 18: Controls structure

CHAPTER 4

CONTROL, LOCALIZATION AND MAPPING

This chapter explores techniques to concurrently estimate in real time the structure of the surrounding perceived by moving sensors and with robot's state estimation and mapping algorithm EKF-SLAM, and the PID to control the motion of the robot Further, the pseudocode presented defines the structure of the algorithm and the flowcharts breaks down the structure of the primary functions used in the algorithm.

4.1. Pseudocode

```
Set Goals
Get sensors data from all robots
while final goal is not reached
        Check for obstacles
        If Obstacle does not exist
                Run EKF (Make State prediction and update coordinates)
                Run Robot
        Else
                Calculate Rebound Angle
                Run PID (Get updated coordinates)
                Run Robot
        End
        Update sensor data from all robots
End
```

4.2. Flowcharts

Below are the flowcharts for the robot's functionality.

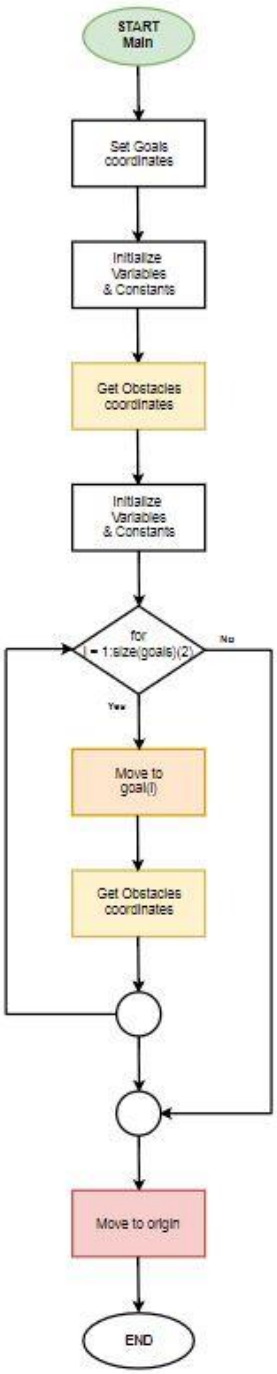
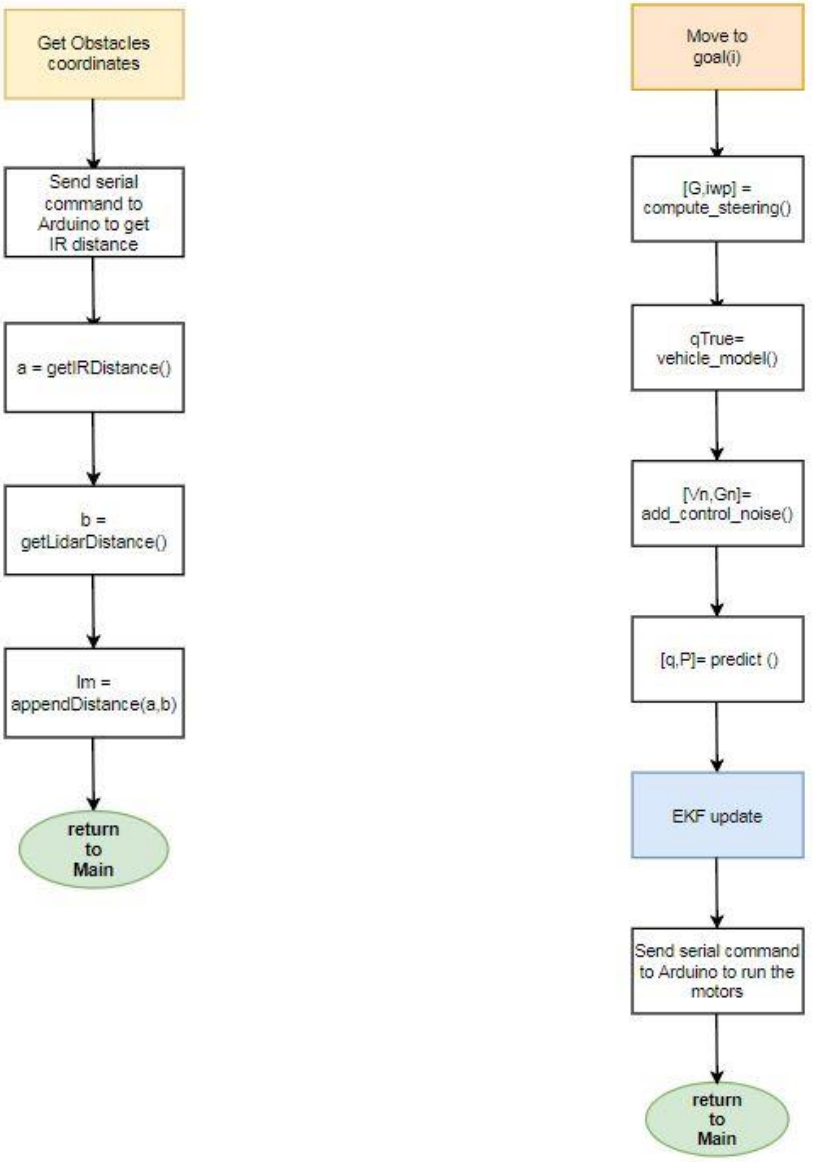
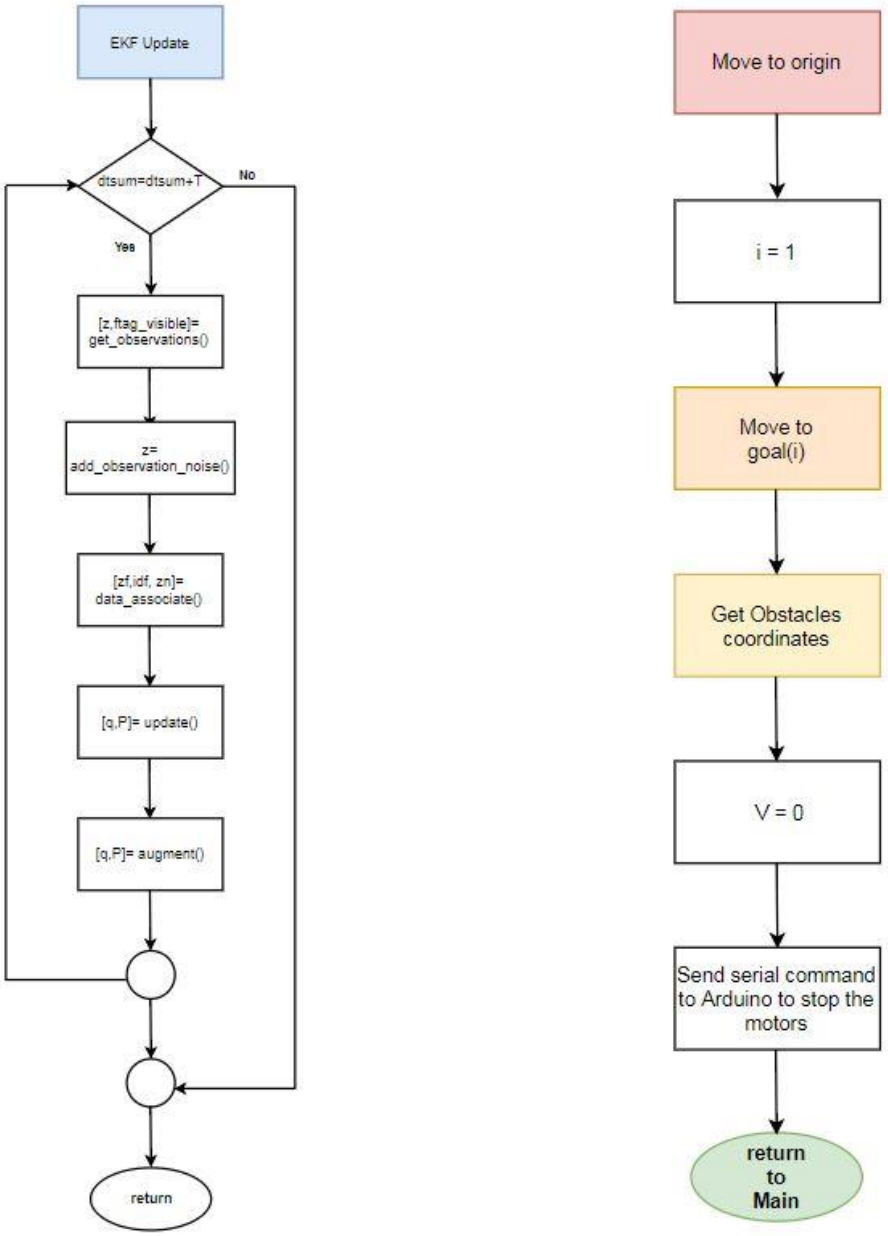


Figure 19: Main System Flowchart



(a) (b)

Figure 20: (a) Flowchart to get Obstacle coordinates (b) Flowchart to move the robot to a desired goal



(a) (b)

Figure 21: (a) Flowchart to update coordinates (b) Flowchart to move the robot to the origin

4.3. Control

The algorithm presented in this research uses a dual level control structure to navigate through the dynamic environment. The higher-level controller uses the predicted position, velocity and time computed using EKF-SLAM to maneuver the robot to the next goal. The robot is primarily run using the higher-level controller. But when an obstacle is encountered, the lower level controller turns on.

This lower level controller uses PID Control to navigate the robots. In this case, the robot turns at a rebound angle determined with respect to the current position of the robot. The robot then moves for 1.5 sec at a velocity of 1m/s to go out of the range of the detected obstacle/s. As soon as the robot is out of range, the higher controller switches back on. The lower level considers the robots as points in the 2-D space and therefore, the PID uses unicycle dynamics as given below,

$$\dot{x} = V\cos\theta \qquad \dot{y} = V\sin\theta \qquad \omega = \dot{\theta}$$

where, $\dot{x}$ is the linear velocity in x direction, $\dot{y}$ is the linear velocity in y direction, V is the velocity, θ is angle of rotation of the robot, $\dot{\theta}$ is the angular velocity and ω is the resultant angular velocity

The objective of the PID control is used to drive the robot towards the rebound angle calculated when the robots detect an obstacle.

The error e and error function ê for the PID controller are calculated as follows, with the desired angle θ^d equal to the rebound angle for this research

$$e = \theta^d - \theta$$

$$\hat{e} = \tan^{-1}\left(\frac{\sin e}{\cos e}\right) \qquad \hat{e} \in [-\pi, \pi]$$

The angular velocity ω can, therefore, be written as,

$$\omega = K_P \hat{e} + K_I \int_0^t \hat{e}(\tau).\, d\tau + K_D \frac{d\hat{e}}{dt}$$

Using the control equations presented in this section, the Simulink model can be created as shown in Figure 22.

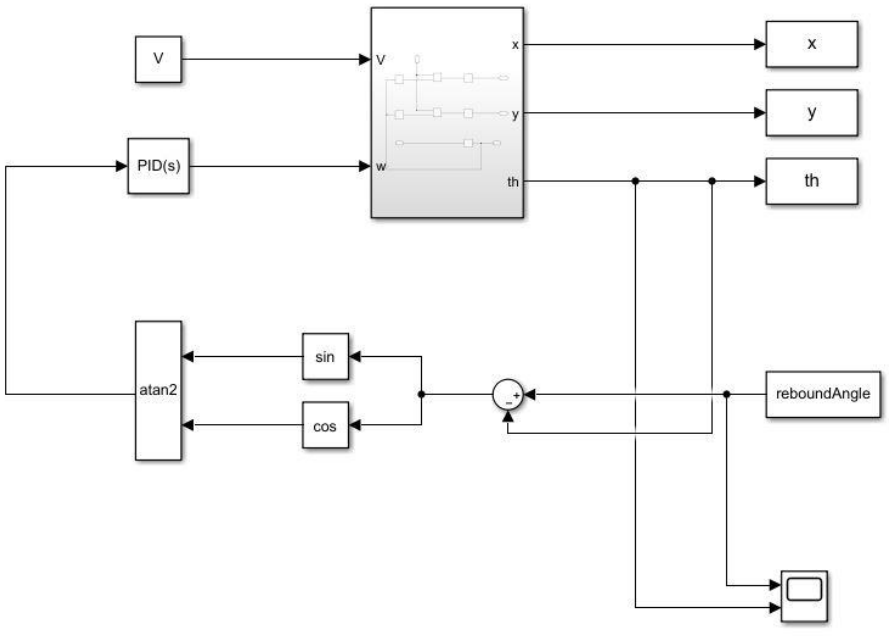


Figure 22: Simulink for PID Control

4.4. Localization and Mapping

In many applications, a model of the environment in which the robot operates is often available. It is often quite advantageous to use this information to plan an optimal path even if some changes in the environment might occurring real time due to the appearance of some dynamic obstacles. Robots often navigate in pre-known environments modeled using different tools such as maps. The Extended Kalman Filter is one way to apply Bayesian estimation techniques to update robot's state at each time step and invoke the state update methods of the Vehicle. The complete history of estimated state and covariance is stored within the EKF object. In EKF-SLAM, the map is a large matrix of sensors and landmarks states which is modeled by a Gaussian variable using the mean and the covariance of the state matrix as shown by,

$$\bar{\mathrm{x}} = \begin{bmatrix} \bar{\mathcal{R}} \\ \bar{\mathcal{M}} \end{bmatrix}$$

where $\mathcal{R}$ is the robot state and $\mathcal{M}$ is the set of all the landmarks, $\bar{x}$ is the mean of the state vector.

$$\mathrm{P} = \begin{bmatrix} \mathrm{P}_{\mathcal{R}\mathcal{R}} & \mathrm{P}_{\mathcal{R}\mathcal{M}} \\ \mathrm{P}_{\mathcal{M}\mathcal{R}} & \mathrm{P}_{\mathcal{M}\mathcal{M}} \end{bmatrix}$$

here, P is the covariance matrix. Therefore, the map is represented as $\{x, \mathrm{P}\}$ and gets updated at every timestep. The generic time-update function is dependent on the state vector x, the control vector u, and the perturbation vector n. It can be defined as,

$$\mathrm{x} \leftarrow \mathrm{f}(\mathrm{x}, \mathrm{u}, \mathrm{n})$$

The sensors return measurements about landmarks relative to the vehicle's location and is represented by a Sensor object. This map is maintained by the EKF through the processes of prediction and correction. The EKF uses the sensor data to collect its surrounding landmark information and predict the robot's position and correct the previously made observations for the landmarks on the map. The EKF prediction step is written as,

$$\bar{x} \leftarrow f(\bar{x}, u, 0)$$

$$P \leftarrow F_x P F_x^T + F_n N F_n^T$$

where the Jacobian matrices are, $F_x = \frac{\partial f(\bar{x},u)}{\partial x}$ and $F_n = \frac{\partial f(\bar{x},u)}{\partial n}$, and N is the covariances matrix of the perturbation n.

In SLAM, the time-variant robot $\mathcal{R}$ is represented as,

$$\mathcal{R} \leftarrow f_{\mathcal{R}}(\mathcal{R}, u, n)$$

$$\mathcal{M} \leftarrow \mathcal{M}$$

here the first equation defines the motion model and as the largest part of the map is invariant due to motion, the sparse Jacobian matrices are given as,

$$F_x = \begin{bmatrix} \frac{\partial f_{\mathcal{R}}}{\partial \mathcal{R}} & 0 \\ 0 & I \end{bmatrix}, F_n = \begin{bmatrix} \frac{\partial f_{\mathcal{R}}}{\partial n} \\ 0 \end{bmatrix}$$

EKF also has an extra step of landmark initialization, where newly discovered landmarks can added to the map. These observation are given as,

$$y = h(x) + v$$

where y is the noisy measurement, x is the full state, h() is the observation function and v is the measurement noise.

Landmark initialization is performed by inverting the observation function and using it and its Jacobians to compute, from the sensor pose and the measurements, the observed landmark state and its necessary co- and cross-variances with the rest of the map. The EKF corrections are then appended to the state vector and the covariances matrix, as shown below,

$$\bar{z} = y - h(\bar{x})$$

$$Z = H_x P H_x^T + \mathcal{R}$$

$$K = P H_x^T Z^{-1}$$

$$\bar{x} \leftarrow \bar{x} + K\bar{z}$$

$$P \leftarrow P - KZK^T$$

here, the Jacobian $H_x = \frac{\partial h(\bar{x})}{\partial x}$ and R is the covariances matrix of the measurement noise. The first two equations represent the mean and the covariance matrix for the newly

discovered landmarks, respectively. The third equation defined the Kalman gain, K. Finally, the last two equations constitute the filter update.

In SLAM, observations occur when a measure of a particular landmark is taken by any of the robot's embarked sensors. The observation function for the observed landmark I is written as,

$$y_I = h_i(\mathcal{R}, S, \mathcal{L}_i) + v$$

where $H_{\mathcal{R}} = \frac{\partial h_i(\bar{\mathcal{R}},S,\bar{\mathcal{L}}_i)}{\partial \mathcal{R}}$ and $H_{\mathcal{L}_i} = \frac{\partial h_i(\bar{\mathcal{R}},S,\bar{\mathcal{L}}_i)}{\partial \mathcal{L}_I}$ represents the Jacobian matrices for EKF-SLAM.

Therefore the correction equations for EKF-SLAM are,

$$\bar{z} = y_i - h_i(\bar{\mathcal{R}}, S, \bar{\mathcal{L}}_i)$$

$$Z = [H_{\mathcal{R}} \quad H_{\mathcal{L}_i}] \begin{bmatrix} P_{\mathcal{R}\mathcal{R}} & P_{\mathcal{R}\mathcal{L}_i} \\ P_{\mathcal{L}_i\mathcal{R}} & P_{\mathcal{L}_i\mathcal{L}_i} \end{bmatrix} \begin{bmatrix} H_{\mathcal{R}}^T \\ H_{\mathcal{L}_i}^T \end{bmatrix} + \mathcal{R}$$

$$K = \begin{bmatrix} P_{\mathcal{R}\mathcal{R}} & P_{\mathcal{R}\mathcal{L}_i} \\ P_{M\mathcal{R}} & P_{M\mathcal{L}_i} \end{bmatrix} \begin{bmatrix} H_R^T \\ H_{\mathcal{L}_i}^T \end{bmatrix} Z^{-1}$$

$$\bar{x} \leftarrow \bar{x} + K\bar{z}$$

$$P \leftarrow P - KZK^T$$

CHAPTER 5

SIMULATIONS

The simulation was performed using two types of static obstacles scenario. In the first case, the obstacles were randomly generated and for the second case, the obstacle was modified to be a circular wall with one opening was designed to demonstrate a complex situation for the robots to maneuver around. For simulation purposes, the robots are considered as a point and the obstacles detected are considered as circular objects with radius equal to the diameter of the robots being considered. This is done so that the robot can move in between two obstacles without any interference. The simulation therefore concludes that the algorithm can be successfully integrated with any hardware system. The hardware implementation and its results are discussed in Chapter 6. Figure 23 presents an shows the robots moving from the origin at (0,0) to three predetermined goals and returning back.

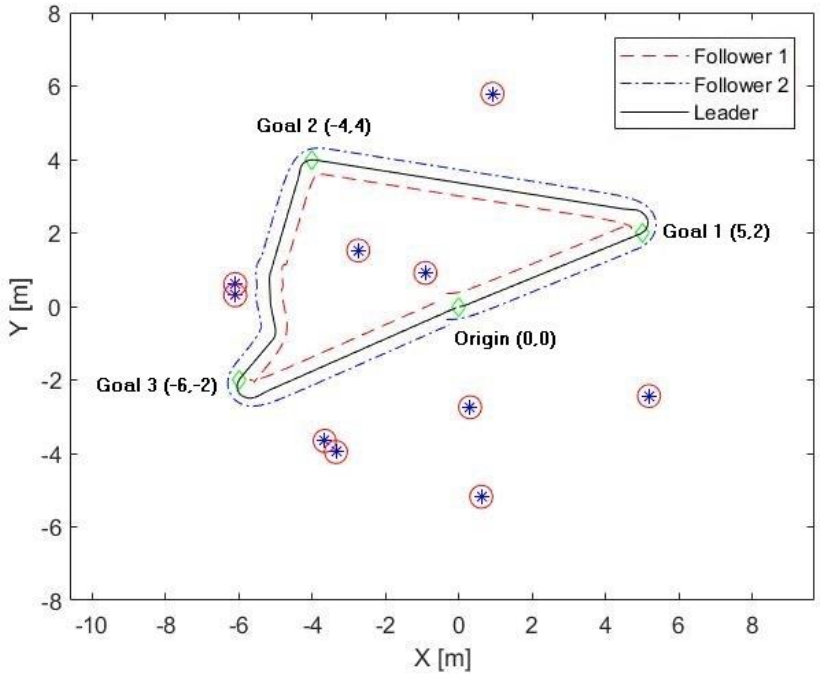


Figure 23: Simulated path for Random Obstacles

The obstacles shown in Figure 23 are randomly generated and the minimum distance for the robot to detect the obstacle and to change its direction is 1 m for the Leader and 0.5 m for the Followers from the obstacle circumference.

From Figure 24 and 25, it can be determined that the relative position of the followers remain fairly constant (~0.5±0.006m). The peaks appearing in the plot can be attributed to the turning motion of the robots. The peak with the flat top at 0.5 m, in Figure 24, indicates that the robots have encountered an obstacle and the PID controller, discussed in Section 4.3, initiates the process of moving the robot out of the obstacle range.

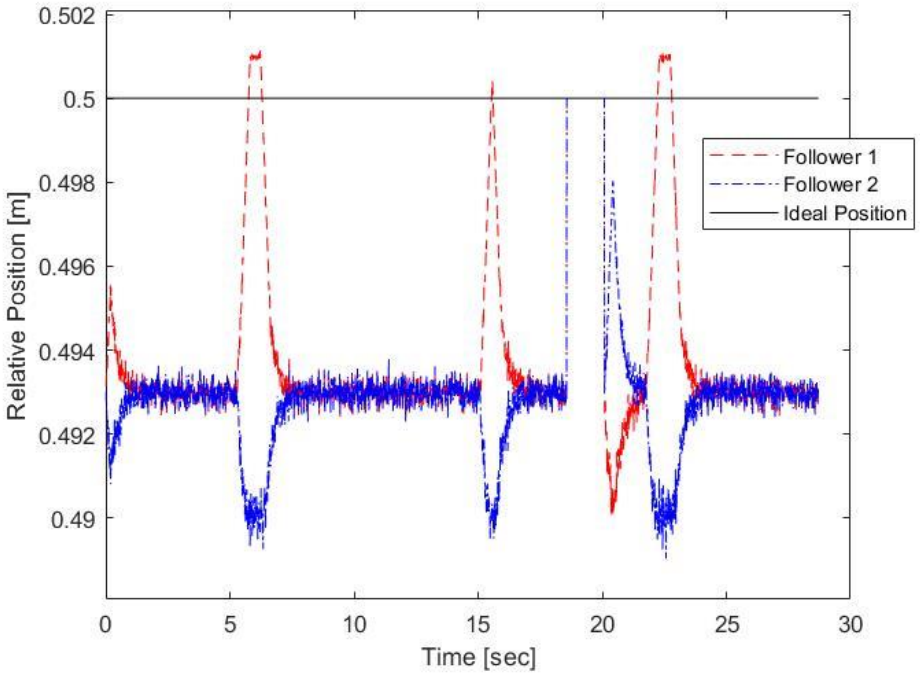


Figure 24: Relative Position of the Followers with respect to the Leader for Random Obstacles

Similarly, the flat peak, in Figure 25, denotes that there is no error in the relative positions between the Leader and Followers. The stability of the formation can also be seen in Figures 24 and 25, where the Followers maintain their distance of 0.5 m from the Leader. Although there are some minor fluctuation in the relative position while the robots are turning from an obstacle, it is enough to maintain a safe space for the robots not to collide.

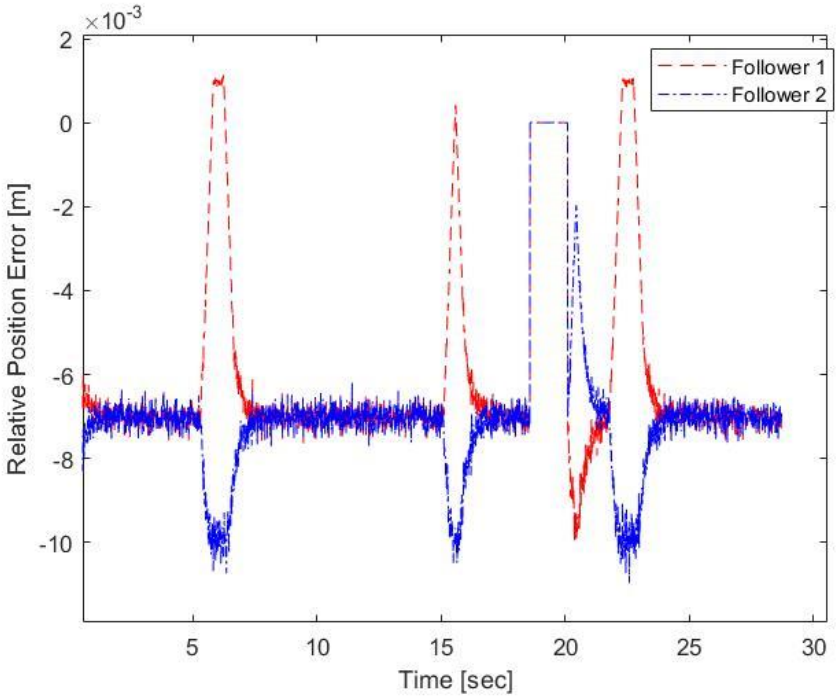


Figure 25: Relative Position Error for Random Obstacles

The plot given by Figure 26, compares the Linear Velocity at which the robots are running. It can be deduced that the velocity difference from the given velocity is almost negligible (~0.05 m/s). Also, the constant velocity line at around 18 sec, can be attributed to the robot detecting an obstacle and the PID. Also, the Velocity error plot shown in Figure 27, has a gap at 18 sec. The error at the gap is zero as the PID runs the motors at the constant velocity of 1 m/s.

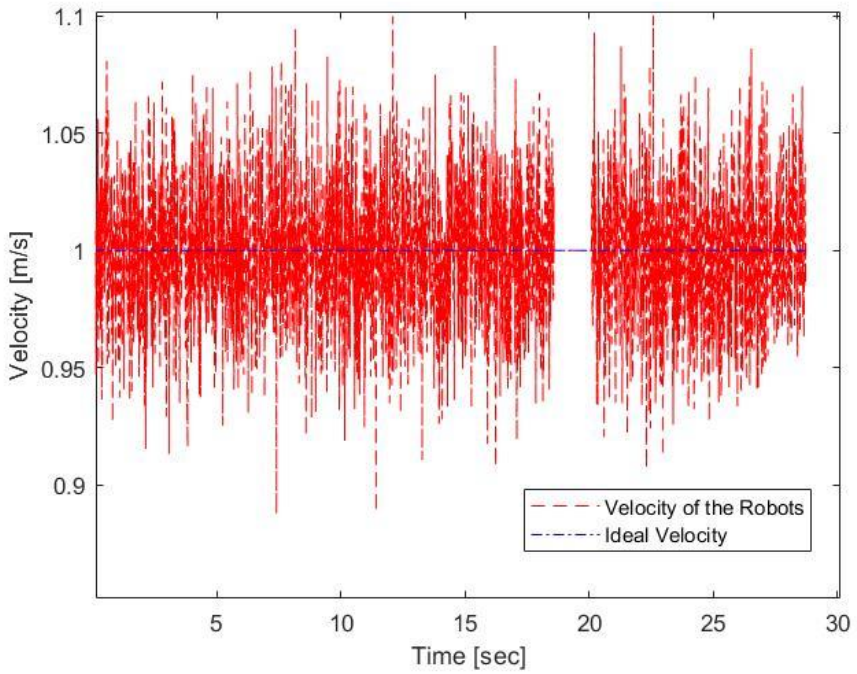


Figure 26: Linear Velocity for Random Obstacles

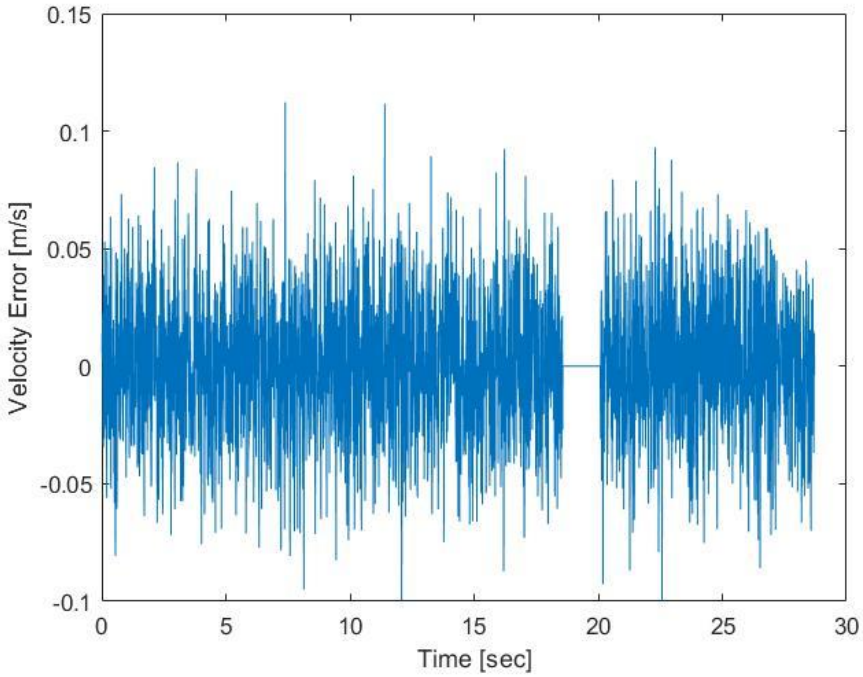


Figure 27: Velocity Error for Random Obstacles

After getting promising results from the first simulation, the second simulation was made complex so that the sensing capability and the effectiveness of the rebound angle can be tested. This was done by conducting the simulation with a circular wall surrounding the robots with only one exit as shown in Figure 28.

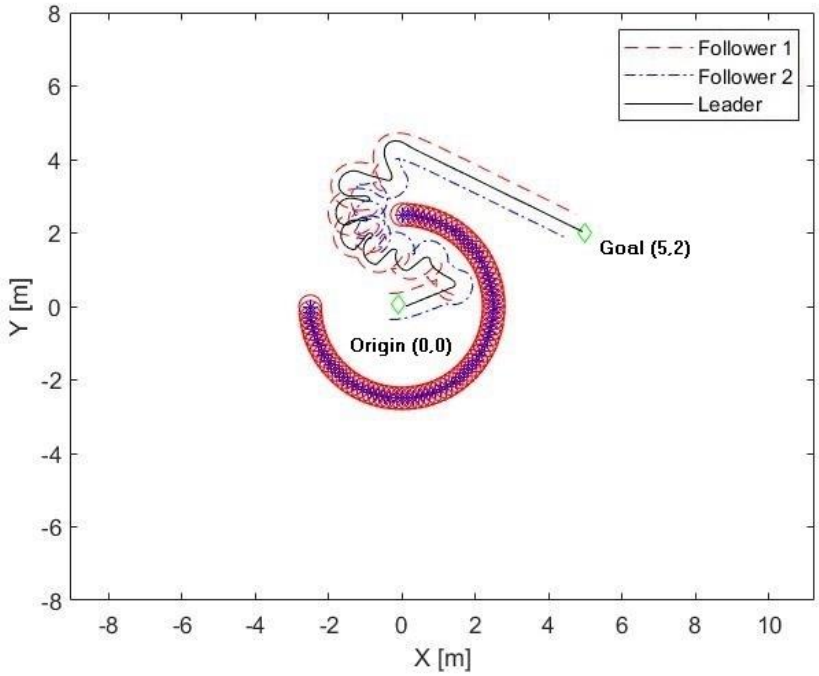


Figure 28: *Simulated path for Complex Obstacles*

As it can be seen from Figure 28, that the robots successfully reach their goal. Further, looking at the relative position (Figure 29) and its error (Figure 30), we can deduce that the plots behave very similar to the aforementioned test, stating a relative position error of about 0.006 m. Multiple obstacles can be confirmed from the plots as they have flat peaks, at 0.5 m, each time the robots encounter an obstacle.

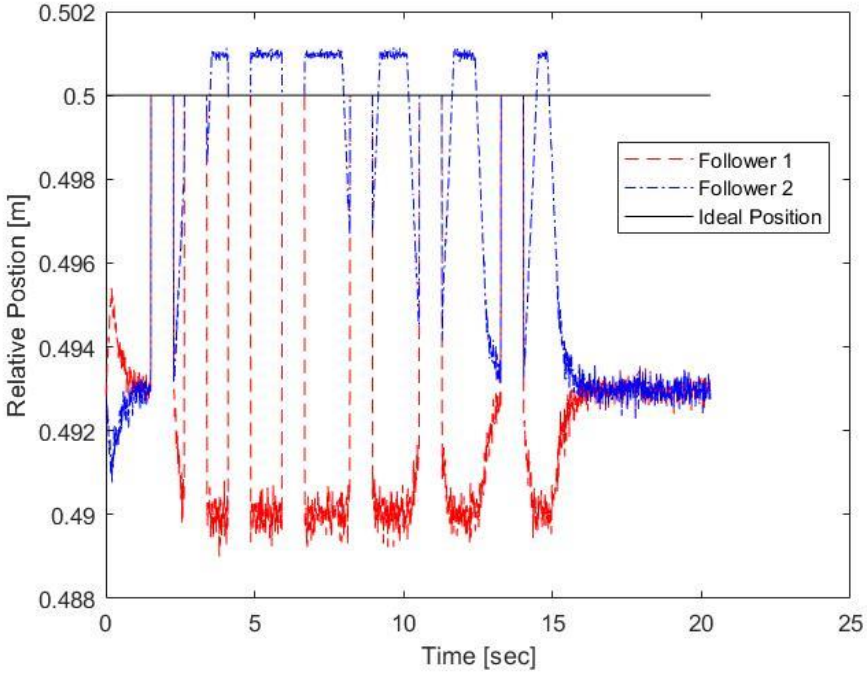


Figure 29: Relative Position of the Followers with respect to the Leader for Complex Obstacles

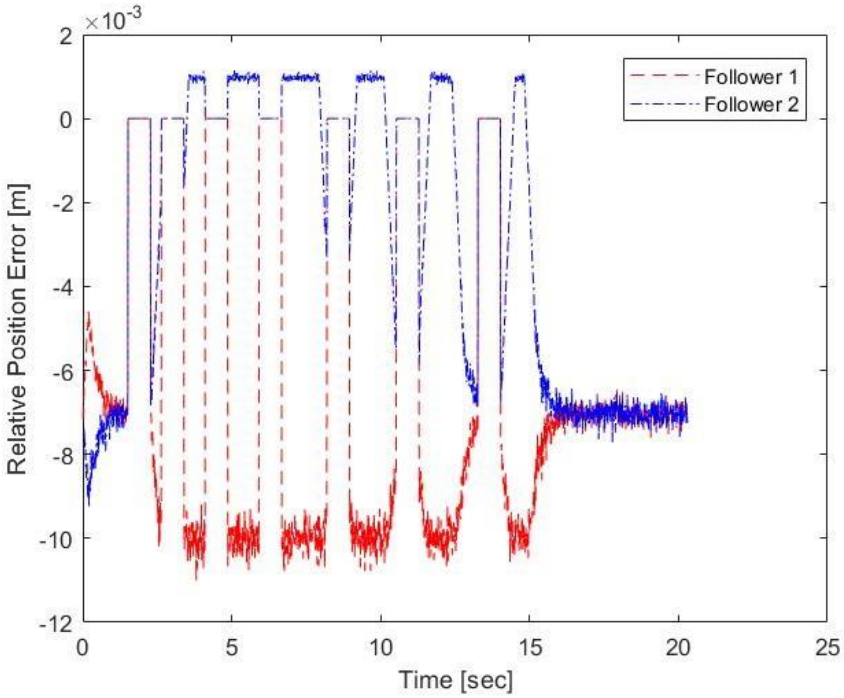


Figure 30: Relative Position Error for Complex Obstacles

Figures 31 and 32, we can see similar gaps as in Figure 26 and 27. These gaps can also be attributed to the PID controller which drives the robots at a constant velocity of 1 m/s.

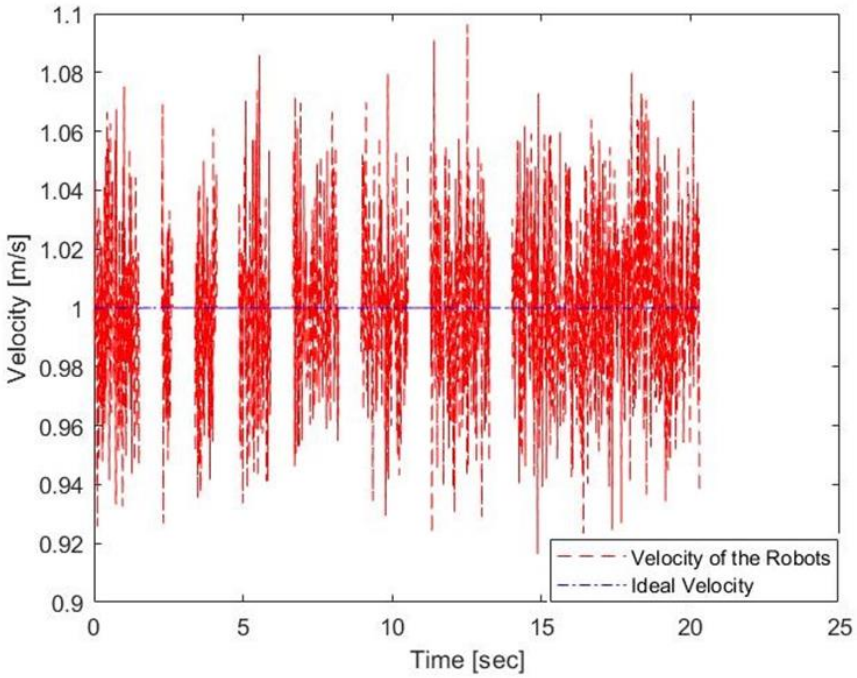


Figure 31: Linear Velocity for Complex Obstacles

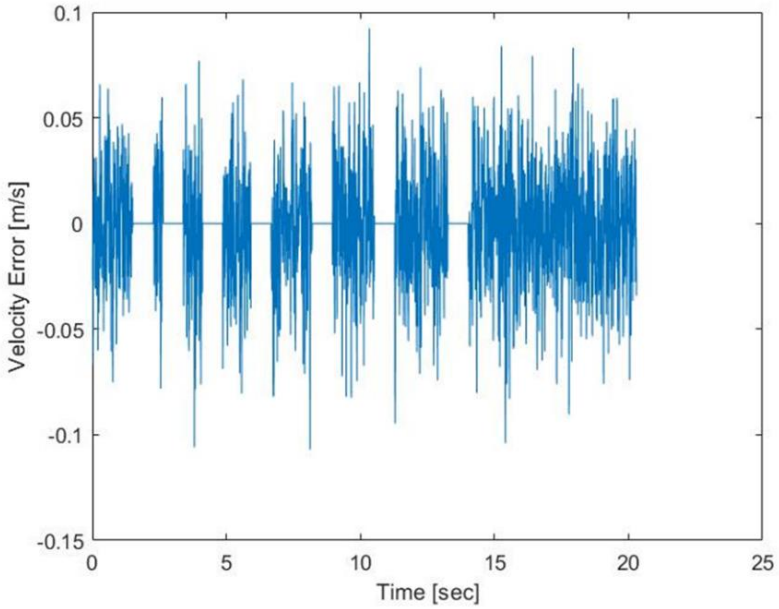


Figure 32: Velocity Error for Complex Obstacles

CHAPTER 6

RESULTS AND DISCUSSION

In this section, we will validate results by implementing the Hardware setup discussed in Chapter 3. Four tests were performed. The first and the second tests, involves a singles robot to go to two and three goals respectively. The first test was an open loop setup where the robots moves from the origin to the final goal and stops. For the second test, a closed loop setup was used, so that the robot returns back to the origin. The third test executed a single robot along with a simulated follower to avoid the obstacles detected by the robots. Finally, the last test implemented two robots to move in a formation from the origin to a defined goal.

The simulation was conducted by changing the ideal linear velocity to 4 m/s for the first two experiments and 0.5 m/s for the third experiment. The tests 1, 2 and 4, were done without considering the existence of an obstacle in the field. Also, for the tests 1, 2, and 3, the follower robots were not implemented for focusing the tests exclusively to path following (for tests 1 and 2) and obstacle avoidance (for test 3). To accommodate the dimensions of the robots, the obstacle detection range, for test 3, was decreased to 0.5 m for the Leader and 0.3m for the simulated Follower. Also, the Lidar used for test 3, was used to map the room before the simulation started to provide the map of the room to the robot. All hardware simulated results used the control, localization and mapping algorithms that has been previously discussed in Chapter 4.

For the first test, the path of the robot, shown in Figure 33, shows that the robot reaches its goal. The curves at the turning points denotes that the turning was smooth, not abrupt.

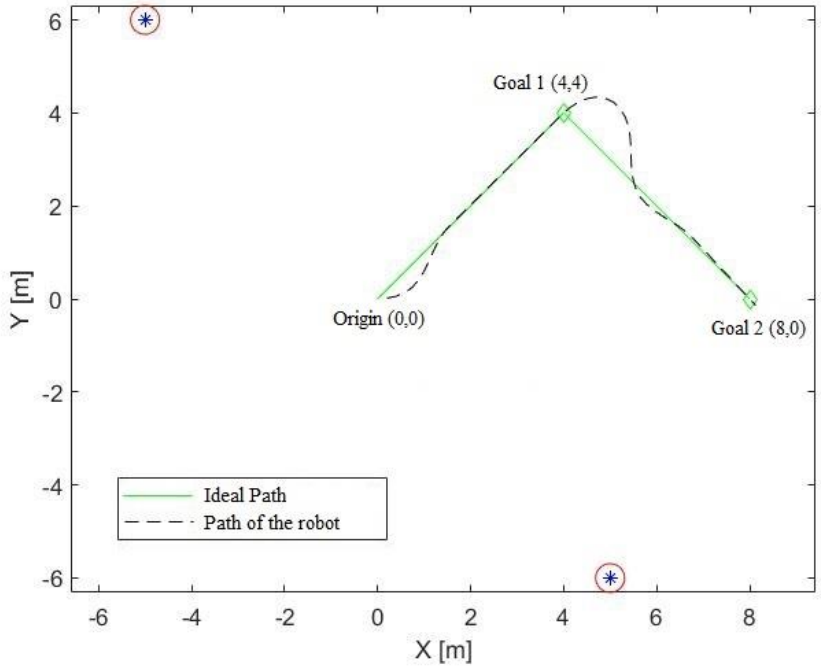


Figure 33: Path following by a Single Robot in an Open Loop

However, the graph doesn't represent the motion of the body, but only the position of its center. In real-time, one can see the robot rotating while translating during its motion. This movement can be attributed to the robot's three omni-directional wheels, which adds an angular velocity to the robot while its moving.

From Figure 34, it can be seen that the velocity of the robot, closely follows the ideal velocity of 4 m/s. The plot also depicts that the robot eventually slows down as it moves closer towards the last goal before coming to a halt. The error, shown in Figure 35, between the ideal velocity and the velocity of the robot is also very negligible (~ -0.05 to 0.07 m/s) and therefore, doesn't significantly affect the motion of the robot.

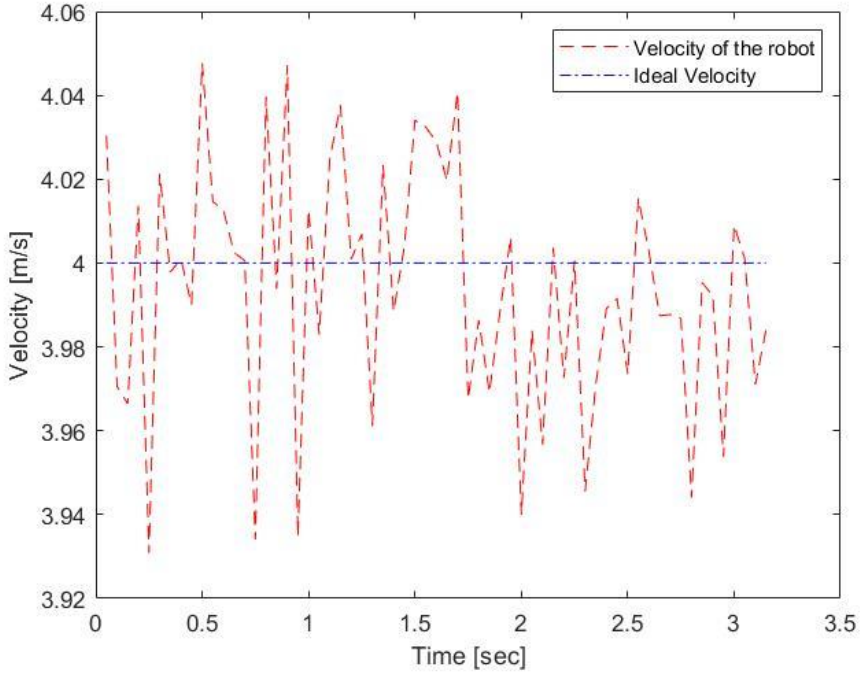


Figure 34: Velocity of a Single Robot in an Open Loop

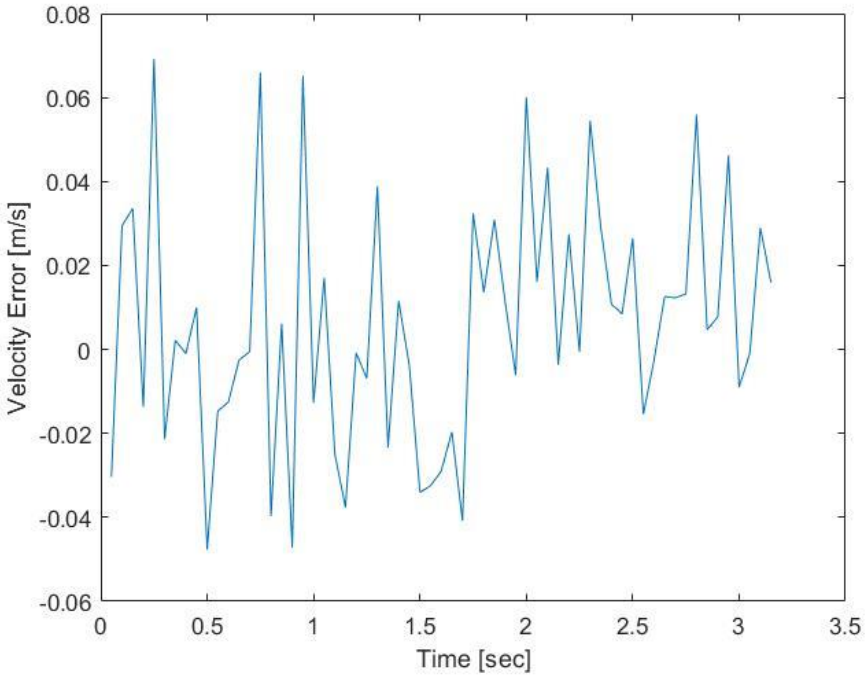


Figure 35: Velocity Error of a Single Robot in an Open Loop

For the second test, the path of a square (shown in Figure 36) was given to the robot. The only difference in this test, from the previous test, is that the robot returns back to its origin after reaching its last goal.

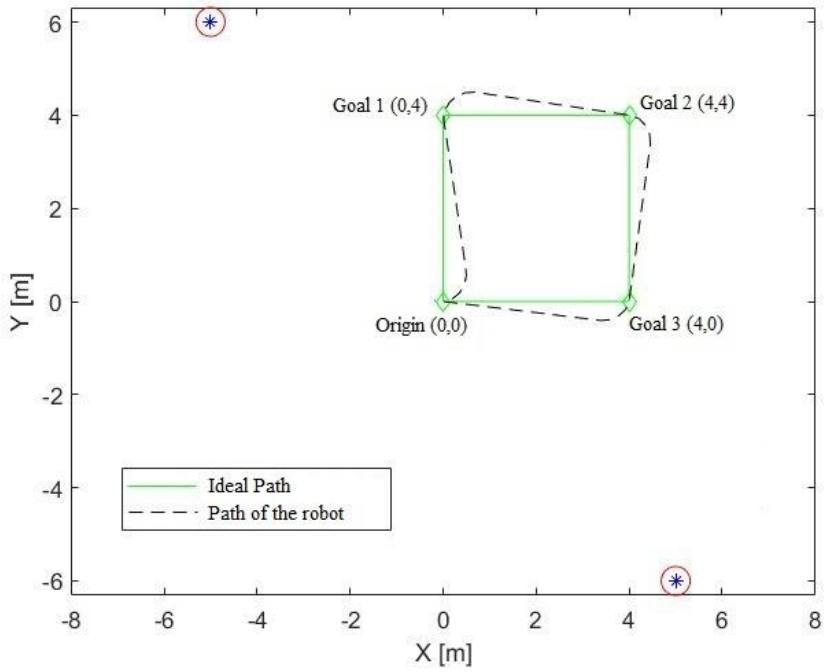


Figure 36: Path following by a Single Robot in a Closed Loop

All the parameters were kept the same from the previous test. The only notable difference can be seen in Figure 37, where the velocity can be seen moving in a wave like motion with almost same amplitude. This can be attribute to the robot going from one equi-distant point of the square to another. Also, from Figure 38, it can be seen that the error between the velocity of the robot and the ideal velocity are very insignificant with an average value ranging from -0.03 m/s to 0.03m/s. This negligible error doesn't affect the linear velocity of the robot which has a much higher magnitude.

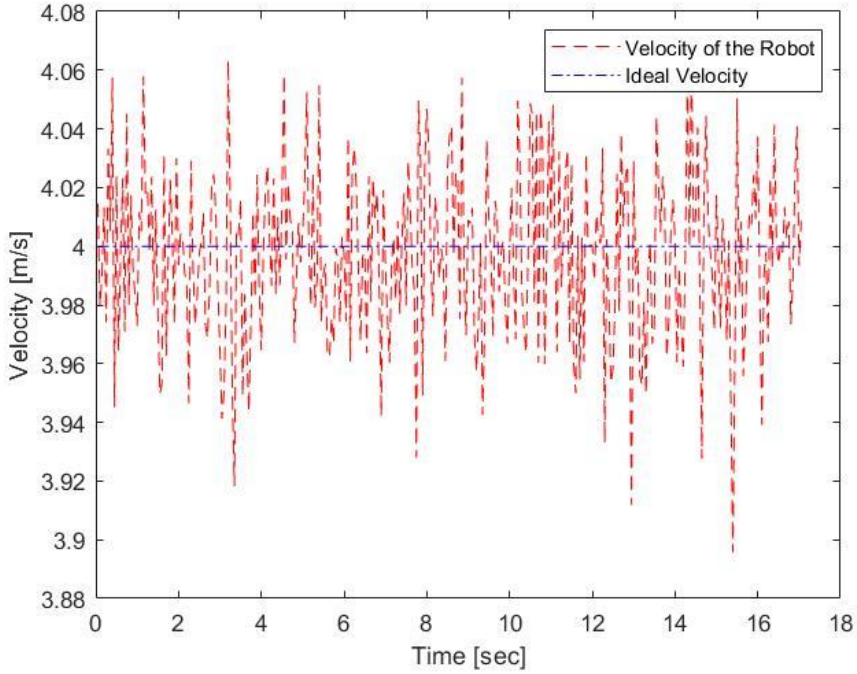


Figure 37: Velocity of a Single Robot in a Closed Loop

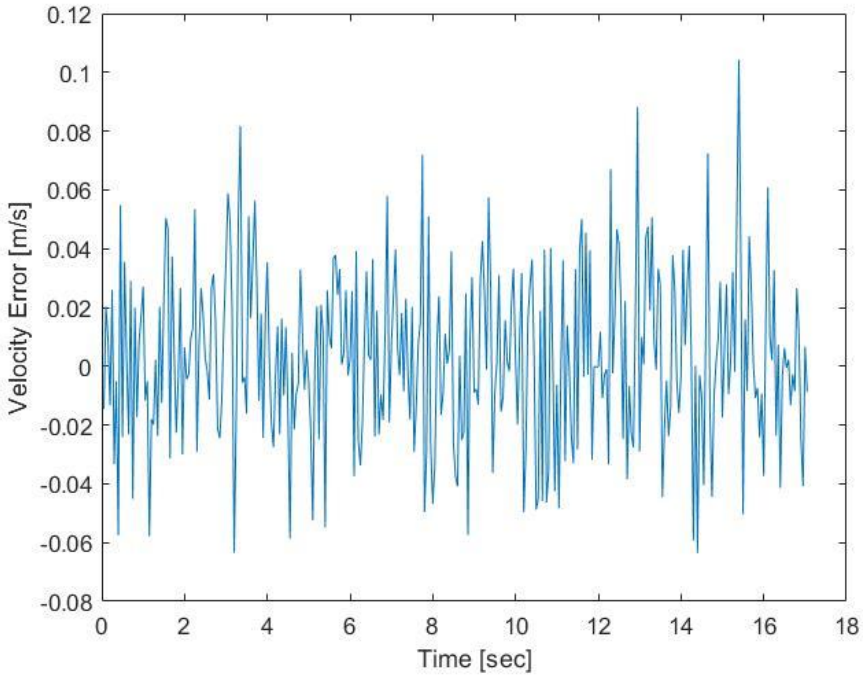


Figure 38: Velocity Error of a Single Robot in a Closed Loop

The third test was implemented by using the Lidar to map the environment of the robot, as seen in Figure 39. The Leader robot moves along with the simulated Follower from the origin to the goal. As it can be seen in the figure below, the robots encounter an obstacle and turns around to avoid it. The robots then move straight and smoothly to its final goal.

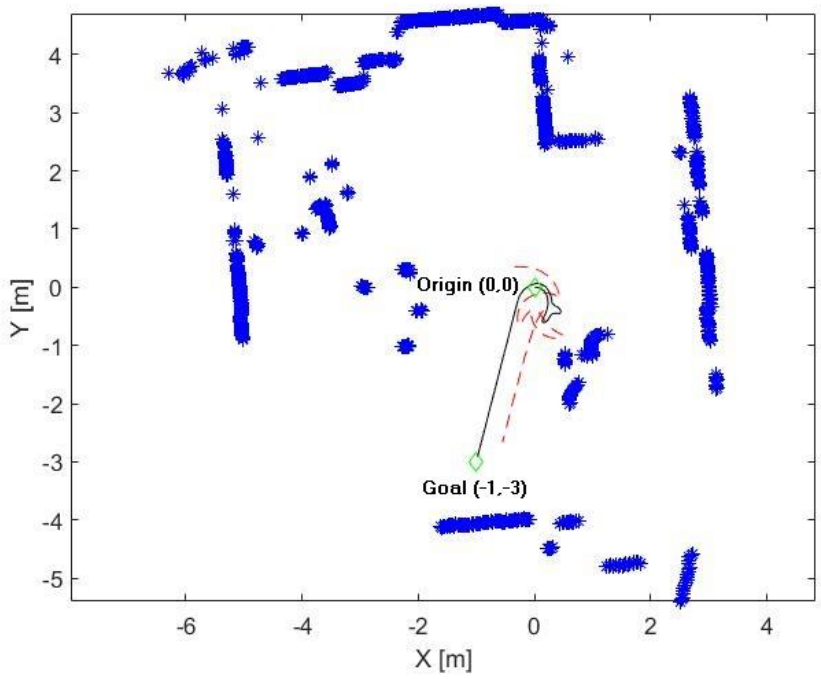


Figure 39: Leader-Follower Path following with Obstacles

The distance between the Leader and the Follower can be seen in Figure 40. From the plot it can be deduced that like the simulated tests, the distance equalizes to the ideal distance, 0.5 m for this case. Also, it can be seen that the distance approaches to a steady value as soon as the robots move straight. Further, the velocity plot shown in Figure 41, also shows similar attributes to the simulated tests, as gaps where the Ideal Velocity (0.5 m/s) equals to the Velocity of the robot.

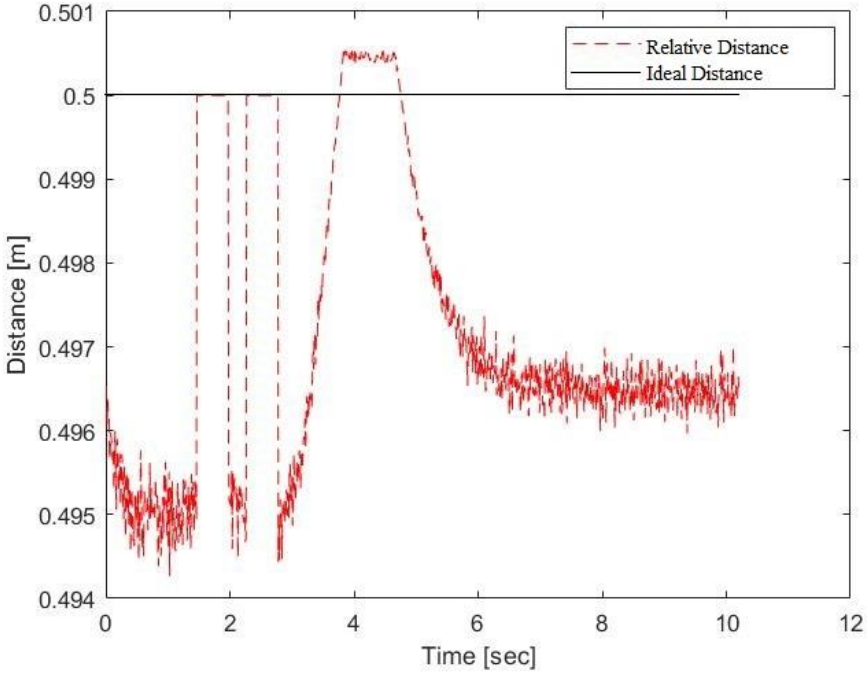


Figure 40: Relative distance for Leader-Follower with Obstacles

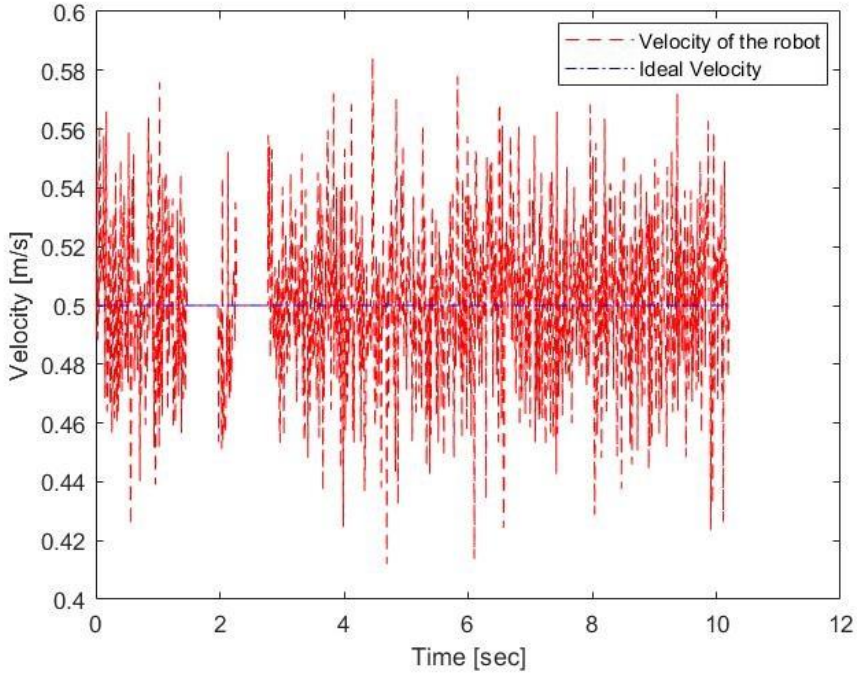


Figure 41: Velocity for Leader-Follower with Obstacles

CONCLUSION AND FUTURE OBJECTIVES

This paper introduces the idea of an efficient algorithm for mapping and obstacle avoidance for robots on a small scale. The presented solution generates a continuous global map and simultaneously updates the robot's location on the map. The higher-level control works accurately and can reach the goals with an accuracy of ± 0.006 m. Also, the lower-level algorithm effectively detects the obstacle from a distance of 1 m and follows the flawlessly calculated rebound angle to avoid any static or dynamic obstacle/s. Although the algorithm works well in a small scale, it has the potential to be upgraded very easily to a more complex algorithm using very high-quality sensors and a more robust robot.

Some key achievements of this research are:

- The robot is able to follow its given path
- The IR sensors and the Lidar are able to detect the obstacle/s from a distance of 1m
- The calculated rebound angle effectively predicts turn angle for the robot to avoid the obstacle
- The lower-level algorithm smoothly follows the rebound angle

Some of future objectives can be

- To upgrade to a higher quality sensor with a better range
- Add more robots with different or unconstrained formation
- To replace the robot chassis to robots, like Pioneer A3, which can work in a rugged terrain
- Extend the functioning capabilities of the robot
- To upgrade serial modules or use API protocols

REFERENCES

[1] Matignon, L., and Simonin, O. (2018, July). *"Multi-Robot Simultaneous Coverage and Mapping of Complex Scene - Comparison of Different Strategies"*. AAMAS '18 Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (pp. 559-567). AAMAS.

[2] Tardós, J., Aragues, R., Sagüés, C., and Rubio, C. (2018, March). *"Simultaneous Deployment and Tracking Multi-Robot Strategies with Connectivity Maintenance"*. Sensors, vol. 18, no. 3, pp. 927.

[3] Manjanna, S., Li, A.L., Smith, R.N., Rekleitis, I., and Dudek, G. (2018, May). *"Heterogeneous Multirobot System for Exploration and Strategic Water Sampling"*. 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 2018, pp. 1-8.

[4] Baede, T.A. (2006). Motion control of an omnidirectional mobile robot.

[5] Rojas, R., and Forster, A.G. (2006). Holonomic Control of a robot with an omnidirectional drive.

[6] Barua, R., Mandal, S., and Mandal, S. (2015, November). Motion Analysis of A Mobile Robot With Three Omni-Directional Wheels. International Journal of Innovative Science (IJISET), Engineering & Technology, Vol. 2 Issue11 (pp. 644-646).

[7] Maalouf, E. (2005, June). *"Nonlinear control of wheeled mobile robots"*.

[8] Mojtahedzadeh, R. (2011). *"Robot Obstacle Avoidance using the Kinect"*.

[9] Susnea, I., Minzu, V., Ochoa-Luna, C., and Vasiliu, G. (2009, December). *"Simple, Real-Time Obstacle Avoidance Algorithm for Mobile Robots"*. 8th WSEAS International Conference: Recent Advances in Computational Intelligence, Man-Machine Systems and Cybernetics (pp. 24-29). WSEAS.

[10] Guo, J., Lin, Z., Cao, M., and Yan, G. (2010, July). *"Adaptive Leader-Follower Formation Control for Autonomous Mobile Robots"*. 2010 American Control Conference (ACC), Baltimore, MD, pp. 6822–6827.

[11] Yu, W., Chen, G., and Cao, M. (2010, June). *"Distributed leader-follower flocking control for multi-agent dynamical systems with time-varying velocities"*. Systems & Control Letters, vol. 59, no. 9, pp. 543-552.

[12] Sola, J. (2014, October). *"Simultaneous localization and mapping with the extended Kalman Filter"*.

[13] Burgardt, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. *"Collaborative Multi-Robot Exploration"*. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2000.

[14] Nieto-Granda, C., Rogers, J.G. III., and Christensen, H. I. *"Coordination strategies for multi-robot exploration and mapping"*. The International Journal of Robotics Research Vol. 33(4), pages 519–533, 2014.

[15] Mehrjerdi, H. (2010, December). *"Control and Coordination for a Group of Mobile Robots in Unknown Environments"*.

[16] Yan, Z., Jouandeau, N., and Cherif, A.A. "*A Survey and Analysis of Multi-Robot Coordination"*. International Journal of Advanced Robotic Systems, 2013.

APPENDIX

IR SENSOR CALIBRATION DATA

cm	V	Sensor Measurement	Error
25	463	25.61672565	2.466903
30	384	31.11883819	3.729461
35	349	34.37078124	1.797768
40	302	39.95035355	0.124116
45	271	44.71361522	0.636411
50	244	49.8703514	0.259297
55	220	55.54030492	0.982373
60	204	60.07758598	0.12931
65	191	64.3358652	1.021746
70	180	68.4296695	2.243329
75	167	73.97802245	1.362637
80	156	79.41056124	0.736798
85	147	84.47298084	0.620023
90	139	89.53492115	0.516754
95	132	94.47805833	0.549412
100	127	98.3494596	1.65054
105	119	105.2347098	0.223533
110	114	110.0390424	0.035493
115	107	117.5354079	2.204703
120	104	121.0634859	0.886238
125	100	126.1037053	0.882964
130	96	131.5726935	1.209764
135	92	137.5271703	1.871978
140	91	139.0992536	0.64339
145	86	147.5195091	1.737592