

ROAD BOUNDARY AND OBSTACLE DETECTION USING 2D LiDAR

A Project Report Phase – II

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication Engineering

by

1. SANJAY KUMAR S (39130398)

2. LOGESHWARAN R (39130261)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 019**

APRIL – 2023



SATHYABAMA

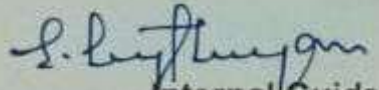
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 019
www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **LOGESHWARAN R (39130261)** and **SANJAY KUMAR S (39130398)** who carried out the project entitled "Road Boundary and Obstacle Detection using 2D LiDAR" under our supervision from NOVEMBER 2023 to April 2023.


Internal Guide

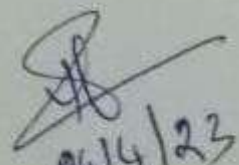
Dr. E. LOGASHANMUGAM, M.E., Ph.D.,

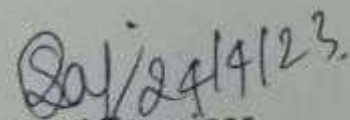


Head of the Department

Dr. T. RAVI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 24/04/2023.


24/4/23
Internal Examiner

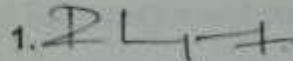

24/4/23
External Examiner

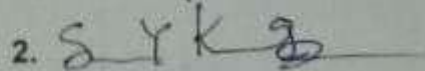
DECLARATION

We LOGESHWARAN R (39130261) and SANJAY KUMAR S (39130398) hereby declare that the Project Report entitled "Road Boundary and Obstacle Detection using 2D LiDAR" done by us under the guidance of Dr. E. LOGASHANMUGAM, M.E., Ph.D., Pro Vice Chancellor, Department of Electronics and Communication Engineering at Sathyabama Institute of Science and Technology is submitted in the partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Electronics & Communication Engineering.

DATE: 24/04/2023

PLACE: Chennai

1. 

2. 

SIGNATURE OF THE CANDIDATES

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. We are grateful to them

We convey our thanks to **Dr. N. M. NANDHITHA, M.E., Ph.D., Professor & Dean, School of Electrical and Electronics Engineering** and **Dr. T. Ravi, M.E., Ph.D., Professor & Head, Department of Electronics and Communication Engineering** for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our **Project Guide Dr. E. LOGANSHANMUGAM, M.E., Ph.D., Pro Vice Chancellor**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express our thanks to all Teaching and Non-Teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

Autonomous vehicles are widely applied in varied road conditions for completing unique jobs with various degrees of autonomy. Road boundary and obstacle detection will provide important information for determining safe vehicle pathways for autonomous vehicles. This needs the vehicle to acquire a certain degree of insight about the road conditions and plan appropriately. Precise and robust perception of surrounding objects of interest, including as on-road obstacles, ground surface, curb and ditch, is a key feature for course planning and localisation in autonomous driving. Most of the prior research has explored road boundary and obstacle avoidance, employing several kinds such as computer vision, ultrasonic, and radar. Our suggested technique employs LiDAR (Light Detection and Ranging) sensors to assess the road surface to determine boundaries in organised and unstructured highways. LiDAR sensors are also used to determine the exact distance to the obstacle so that the autonomous vehicle can identify the thing at a set distance and avoid hitting it. The test results reveal that our boundary detection method has the high accuracy and reliability and is able to correctly achieve the boundary detection job, which can satisfy the requirements of real-time system.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	Abstract	v
	List of Figures	ix
	List of Tables	x
1	INTRODUCTION	1
	1.1 History of Autonomous Vehicle	2
	1.2 Advancement in the Industry	3
	1.3 Defining the Level of Automation	4
2	LITERATURE SURVEY	8
	2.1 Interference from Literature Survey	8
	2.2 Problems in Existing Methodology	11
3	REQUIREMENT ANALYSIS	13
	3.1 Hardware Requirements	13
	3.2 Software Requirements	19
	3.3 Risk Analysis of 2D LIDAR	22
	3.4 Feasibility Study of 2D LIDAR over 3D LIDAR	24
	3.5 Feasibility Study of Encoder Motors over Normal Motors	25
	3.6 System Use Case of the Project	26
4	PROPOSED METHODOLOGY	28
	4.1 Functional Block of Proposed Methodology	28
	4.2 Flow Diagram of Proposed Methodology	30
	4.3 Communication Model of Proposed Methodology	31

4.4	Integration of ROS in Proposed Methodology	32
4.5	Slam Technique	33
4.6	Slam Gmapping Algorithm	34
4.8	Project Management Plan	36
4.9	Cost Estimation of Project	37
5	PROJECT IMPLEMENTATION	38
5.1	Robot Design	38
5.2	RP LIDAR A2 M8 Testing	39
5.3	Communication between Master and Slave	40
5.4	Teleop of the Robot	42
5.5	Tick Value Measuremnet	43
5.6	URDF Generation	45
5.7	Mapping of the Environment	46
5.8	Navigation of the Robot	48
6	RESULT AND DISCUSSION	50
6.1	Stimulation Result	50
6.2	Real Time Enivronment Result	53
7	CONCLUSION	57
7.1	Conclusion	57
7.2	Future Work	57
7.3	Research Issues	58
7.4	Implementation Issues	58
	REFEERENCES	59
	APPENDIX A	61
	APPENDIX B	64
	APPENDIX C	66

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	SAE Automation Levels	4
1.2	Defining the Automation	5
3.1	RP LIDAR A2 M8	13
3.2	Encoder Motors	15
3.3	Raspberry Pi 4B	16
3.4	Arduino Mega	16
3.5	Motor Drivers	17
3.6	Batteries	18
3.7	Car Chassis	18
3.8	Arduino IDE	19
3.9	Raspbian OS	19
3.1	Ubuntu 20.04	20
3.11	ROS Melodic	21
3.12	Rviz	21
3.13	Gazebo	22
4.1	Block Diagram	28
4.2	Flow Diagram	30
4.3	Communication Model	31
4.4	Integration of ROS	32
4.5	SLAM Technique	33
4.6	SLAM GMapping Flow Chart	35
5.1	Robot Model	39
5.2	ROS Serial Master Slave	42
5.3	URDF of the Robot	45
6.1	Indoor Environment	50
6.2	Mapped Environment	51
6.3	Robot Mapped to detection with avoid obstacles	51

6.4	Path Planning and Obstacle Avoidance of the Robot	52
6.5	Path Planning and Obstacle Avoidance Simulation in Gazebo Tool	52
6.6	Straight view of the Robot	53
6.7	Side view of the Robot	53
6.8	Real time environment	54
6.9	Mapping of the Environment	54
6.10	YAML Map Image	55
6.11	Navigation Environment	55
6.12	Navigation Visualisation in RViz	56
6.13	Real Time Navigation Output	56

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
3.1	LiDAR Parameters	14
3.2	Comparison of LiDAR with Various Sensors	14
3.3	Encoder Motor Parameters	15
4.1	Project Phase I Management	37
4.2	Project Phase II Management	37
4.3	Cost Estimation	37
5.1	Tick Value Observation	44

CHAPTER 1

INTRODUCTION

Every year around in the globe approximately 1.5 million fatalities caused by traffic accident. At age 18 or normally the new learner who are just learning how to drive, it's worrisome for me to think about the potential accident And what if the Accident happened due to someone other error. On the top of that, since we know that amount of traffic, we have it only cause the useless/unnecessary irritation for practically every 9 individuals out of 10. This was the key cause for selecting our project that is and Autonomous Vehicle which is also known as autonomous vehicle (A vehicle which can operate without Driver or sometimes known as robotic Vehicle) (A vehicle which can run without Driver or also known as robotic Vehicle). It is a vehicle that is capable of detecting its environment and also navigating the input without the aid of person. This sort of vehicle employs variety of technology as Lidar, radar, laser light gps computer vision. Owing to this this sort of vehicle also known as human brain because of the capacity to perceive the environment and also supply the input by itself.

According to study there have been testing and experiment going on to produce self- driving vehicle for over a 40-45 year. This sort of vehicle needs vast variety of technology and infrastructure to function correctly. With the progress of technology, there have been autonomous vehicles. It is an idea which is promoted and backed by Google, and numerous other business. As we know when the Advancement in technology develops the worry for the technology safety is also increases. the key worry is the security of the vehicle. the way we human drive the vehicle is lot different the way robotic brain think as example we state the misunderstanding between vehicle and the passenger we human can drive at our own will to make passenger happy but autonomous vehicle will only take the safest and only route which are supplied to it. it is tough to believe that if the vehicle makes any type of error like crossing at high speed or disregarding signal due of technological failure how it can repair it mistake on its own. And the second most serious issue our that people will never be able to trust the machine as they trust our fellow humans without knowing the logic of the vehicle and how its work. It is

crucial to comprehend the concerns that surround the reasoning that has been applied in the vehicle.

There are many more issues surrounding robotic vehicles. Here are some questions i.e. Would it be feasible to have modifications in the areas where the owner of the driverless vehicle wishes to go? What if I want to alter where I am going? If there is a communication barrier, how can we communicate with the system or vehicle? Since 2016, autonomous vehicles have evolved towards partial autonomy, including technologies that assist drivers remain in their lane, coupled with ACC technology and the ability to self-park. Creators of autonomous vehicles utilise the large quantity of data from many datasets such as image recognition, machine learning and neural networks to construct systems which can operate in a normal environment.

1.1 HISTORY OF AUTONOMOUS VEHICLE

The notion of autonomous or autonomous vehicles has gained traction in the previous decade. Nonetheless, it appears that the notion of developing a vehicle that was capable of navigating itself without human input was envisaged far back in the Middle Ages. Leonardo da Vinci's designs of the self-propelled vehiclet was supposed to be driven by coiled springs with programmed steering and braking capabilities. Clearly since then, technology has improved significantly. Origin of an idea, The earliest reported genuine effort towards developing a driver-less vehicle may be dated to 1925, when Francis Houdina built a radio operated vehicle. The "Phantom Auto," as it was called then, was controlled using radio signals received from another vehicle that drove close behind. Around a decade later, an industrialist by the name of Norman Bel Geddes conceptualised an autonomous highway system whereby vehicles were controlled through electric circuits placed into the pavements. This prototype was successfully tested but didn't take off owing to the enormous financial expenditure necessary. At the same period, the Transport and Road Research Laboratory in the UK imagined a comparable autonomous vehicle and road system that was also successfully tested, and expected to cut road accidents and boost road capacity considerably. Nevertheless, the proposal was cancelled owing to the government's unwillingness to support the initiative.

As early as 1925, Francis Houdina presented a remote-control vehicle named the American wonder, which drove around the streets of Manhattan. In Europe's

early advancements [15] in the 1980s, pioneers worked on autonomous technology for over 40 years, with early vehicles largely partly autonomous. In the year 1986, Ernst Dickens and his team from the University of Munich produced a robotic vehicle that was able to drive totally independently without traffic. In 1987, it extended its speeds up to 60 kilometres per hour. During the same period in the US, Vehicle Negie Mellon University's Navlab was busy constructing a constant sequence of prototypes with ever-increasing capabilities. In 1986, their first autonomous vehicle, Navlab1, accomplished 30 kilometres per hour on the road. Later in 1990, Navlab2, a modified Humvee, functioned independently both off and on-road.

1.2 ADVANCEMENT IN THE INDUSTRY

The Tsukuba Mechanical Engineering Laboratory in Japan produced a concept in 1977 that focused on making the vehicle intelligent rather than on exterior road technology. The technology consisted of a computer that used in-built camera footage to study the surrounding area. This was the first autonomous vehicle capable of travelling at a speed of 20 miles per hour. Later, in the 1980s, a German aerospace engineer employed artificial intelligence to create a prototype- VAMORs- that could self-drive at high speeds. It was made using a Mercedes vehicle and ran on a computer software that passed off data obtained by integrated cameras and sensors. Motivated by this accomplishment, the Eureka Prometheus Project, Europe's biggest R&D effort in autonomous vehicles, began in 1987 with a number of institutions and automotive manufacturers participating. Ernst Dickmanns and his colleagues exhibited the autonomous capabilities of his twin robot vehicles VaMP and VITA-2 by driving 1000 km at 130 km/h on the Paris motorway.

Simultaneously, comparable efforts were being undertaken in America, the most noteworthy of which was the Navlab autonomous vehicle system, which travelled over 3000 miles in 1995 using a supercomputer, GPS receiver, and camera equipment. The DARPA Grand Challenge was established in 2000, promising a million dollars to anybody who could create an autonomous vehicle capable of travelling 150 miles over the Mojave Desert. Regrettably, no one was able to complete this challenge. Yet, Google revealed in 2010 that it has been working on a covert project testing and creating a technology that may help minimise the amount of accidents via autonomous driving. This team included several of the

engineers that worked on vehicles in the DARPA competition. Since then, Google has driven over a million miles and fought to help pass laws making autonomous vehicles legal in four US states. Today, autonomous vehicles pick up and drop off passengers in Pittsburgh, Phoenix, and Boston. Nvidia and Volkswagen just revealed a autonomous chip that combines artificial intelligence to production-ready hardware. This is intended to increase the performance of autonomous vehicles while also including appealing elements such as digital assistants. While autonomous vehicle technology has improved significantly, with vehicles including highly sophisticated driver aid technologies such as Tesla's Autopilot and Cadillac's Super Cruise, it remains to be seen if authorities will authorise the worldwide deployment of autonomous vehicles.

1.3 DEFINING THE LEVEL OF AUTOMATION

Driving automation is not an all-or-nothing approach. Not at all. Automakers are gradually incorporating active safety and autonomous systems into their vehicles. These characteristics are often classified into logical groupings depending on how they integrate acceleration and braking (known as longitudinal control) and steering (called lateral control). Certain elements perform the same job but vary in the degree of human control vs. autonomous system control of the vehicle, corresponding to differing degrees of driving automation.

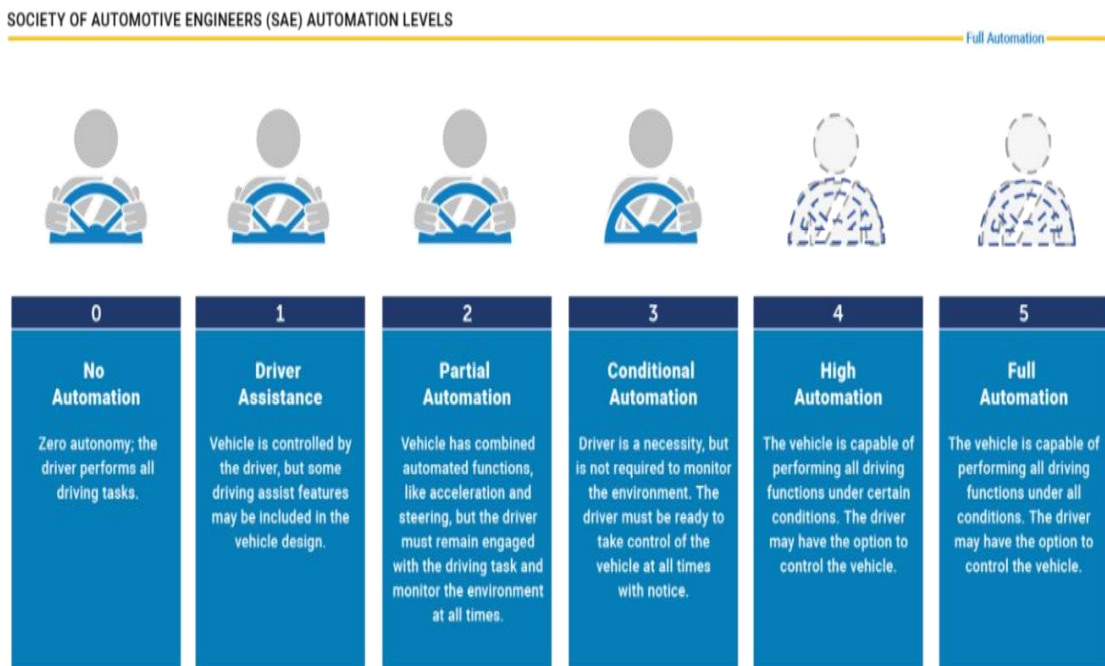


Fig:1.1: SAE Automation Levels

While there are numerous grey regions where characteristics may overlap, the Society of Automotive Engineers (SAE) devised an industry-standard scale from zero to five to depict this continuum. These are the general meanings of those levels:

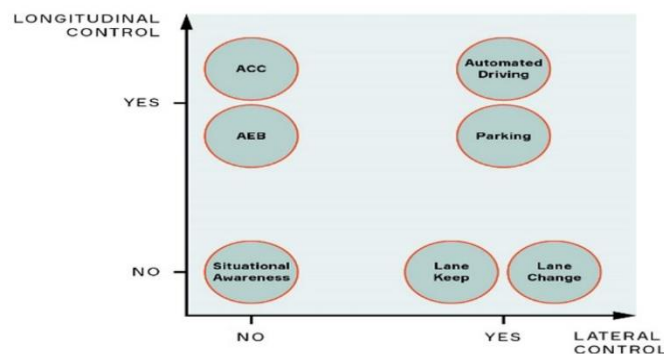


Fig:1.2: Defining the Automation

- **Level 0:** No Automation. The driver is solely responsible for operating the vehicle, executing actions like steering, braking, accelerating or slowing down. Level 0 vehicles may include safety features such as backup cameras, blind spot alerts and collision warnings. Even autonomous emergency braking, which provides severe brakes in the case of an impending accident, is categorised as Level 0 since it does not function over a continuous time.
- **Level 1:** Driver Assistance. At this stage, the autonomous systems start to assume control of the vehicle in specified circumstances, but do not entirely take over. An example of Level 1 automation is adaptive cruise control, which handles acceleration and braking, often in highway driving. Depending on the functionality, drivers are able to lift their feet off the pedals.
- **Level 2:** Partial Automation. At this stage, the vehicle can execute more sophisticated duties that connect steering (lateral control) with accelerating and braking (longitudinal control), owing to a higher understanding of its surroundings. Level 2+: Advanced Partial Automation. Although Level 2+ is not one of the officially recognised SAE levels, it represents an important category that provides advanced performance at a price customers can afford. Level 2+

contains tasks where the vehicle systems are basically driving, but the driver is still expected to monitor the vehicle and be ready to step in if needed. (By comparison, Level 3 marks a substantial technical breakthrough, since it is the first level at which drivers may detach from the task of driving — frequently referred to as “mind off.” In Level 3, the vehicle must be able to safely stop in the case of a malfunction, requiring far more complicated software and hardware.) Examples of Level 2+ include highway assistance or traffic congestion help. The opportunity for drivers to take their hands off the wheel and gaze aside from the road ahead for a few minutes allows for a far more relaxed and pleasurable experience, thus there is tremendous customer demand.

- **Level 3:** Conditional Automation. In Level 3, drivers may detach from the act of driving, but only under specified conditions. Circumstances might be restricted to particular vehicle speeds, road types and weather conditions. However since drivers may apply their concentration to some other work like as glancing at a phone or newspaper this is typically regarded the first entrance point towards autonomous driving. Yet, the driver is supposed to take over when the system asks it. For example, features such as traffic jam pilot mean that drivers can sit back and relax as the system handles it all acceleration, steering and braking. In stop-and-go traffic, the vehicle sends a warning to the driver to reclaim control when the vehicle gets past the traffic bottleneck and vehicle speed rises. The vehicle must also monitor the driver’s status to verify that the driver recovers control, and be able to come to a safe stop if the driver does not.
- **Level 4:** High Automation. At this level, the vehicle’s autonomous driving system is completely capable of monitoring the driving environment and performing all driving operations for normal routes and circumstances described within its operational design domain (ODD) (ODD). The vehicle may inform the driver that it is exceeding its operating limitations if there is, say, an environmental situation that demands a person in control, such as thick snow. If the driver does not answer, it will secure the vehicle automatically.
- **Level 5:** Complete Automation. Level 5-capable vehicles are totally autonomous. No driver is necessary behind the wheel at all. In fact, Level 5 vehicles could not even have a steering wheel or gas/brake pedals. Level 5

vehicles might incorporate “smart cabins” so that passengers can give voice commands to pick a location or control cabin parameters such as temperature or choice of media. In April 2021, the SAE announced an update to their taxonomy to explain that Levels 0-2 are “driver assistance features” since the driver is still extensively engaged with the vehicle operation, while Levels 3-5 are “autonomous driving features. Each degree of automation demands more layers of sensors, as the vehicles progressively undertake duties formerly performed by the driver. For example, a Level 1 vehicle could just have one radar and one camera. A Level 5 vehicle, which must be able to traverse any environment it meets, will need complete 360-degree sensing across many sensor types

CHAPTER 2

LITERATURE SURVEY

Many people done projects on obstacle detection and road boundary detection in different ways. Here we have collected some references. And we came up with a project obstacle avoidance and road boundary detection using 2D LiDAR. Here some of the references related to the obstacle avoidance and road boundary detection.

2.1 INTERFERENCE FROM THE LITERATURE SURVEY

Shan Huang (2022) et al presented the work on A Robust 2D Lidar SLAM Method in Complex Environment. This paper proposes a robust 2D lidar SLAM method for complex environments. The constructed multi-sensor tightly-coupled 2D lidar SLAM framework is suitable for multi-sensor and multi-feature fusion. First, we introduce line landmark tracking and degradation analysis and improve the matching accuracy of the front-end and the stability in the complex environment. The future work is to introduce semantic features to make the mapping process reliable, more accurate and human-friendly [5].

Juan An (2021) et al presented the work on Localization and Navigation Analysis of Mobile Robot Based on SLAM. This paper mainly introduces the color image and depth image provided by the RGB-D sensor, which is based on ORB-SLAM. It expands the sparse image of the original system sealing structure into a dense point cloud image. At the same time, they recommend using an octree structure to construct eight robots. Finally, the algorithm was applied to autonomous mobile robots [10].

Xuexi Zhang (2020) et al presented the work on 2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots. A* is paper studies mapping and path planning for mobile robots in an indoor rescue environment. Combined with path planning algorithm, this paper analyzes the applicability of three SLAM algorithms (GMapping algorithm, Hector-SLAM algorithm, and Cartographer algorithm) in indoor rescue environment. Real-time path planning is studied to test the mapping results. To balance path optimality and obstacle avoidance, A* algorithm is used for global path planning, and DWA algorithm is adopted for local

path planning. Further optimisation needs to be carried out in the mapping algorithms to make them more suitable to the real rescue environment [15].

Sanjyoti Jain (2020) et al presented the work on A Review on Obstacle Avoidance Techniques for Self-Driving Vehicle. In this paper, static as well as mobile obstacles avoidance techniques in known and unknown environment are briefly discussed. This paper discusses sensors used in obstacle avoidance and the algorithms used in the obstacle avoidance [8].

Kai Yan (2020) et al presented the work on Mapless Navigation Based on 2D LIDAR in Complex Unknown Environments. This paper presented a sub-goal seeking approach for local navigation in complex, unknown environments, which can identify gaps and select one of the gaps as a sub-goal to guide the robot to move toward the target. This paper has two main contributions. One is to design a clever gap recognition method, and the other is finding the common defect of the local algorithms and making improvements. It can avoid the robot from being trapped in complex environments so that the local algorithm can complete complex navigation tasks independently and no longer depends on the global algorithms. Therefore, the proposed approach has low computational complexity, which is of great significance to robots with low computing power [14].

Ayesha Iqbal (2020) et al presented the work on Obstacle Detection and Track Detection in Autonomous Cars. This paper helps understanding and categorizing the different techniques used for obstacle detection and tracking as well as the researches that refer to these techniques. The paper provides the advantages and disadvantages of various sensors used in obstacle detection and tracking [6].

Madhavan T.R. (2019) et al presented the work on Obstacle Detection and Obstacle Avoidance Algorithm based on 2-D RPLiDAR. The Lidar system used in the paper is 2D RP Lidar. The algorithm was designed in such a way that obstacle classification and obstacle avoidance are implemented using 2D lidar system which has a single plane of measurement. The data obtained from lidar is filtered, pre-processed and clustered. This paper also proposed a simple obstacle avoidance algorithm using mobile robot. The simulation of both the algorithm is

implemented using MATLAB. Future work will deal with dynamic obstacle avoidance [11].

Fabien Rakotondrajao (2019) et al presented the work on Road Boundary Detection for Straight Lane Lines Using Automatic Inverse Perspective Mapping. An approach to automatically select points for performing the Inverse Perspective Mapping in fitting step of lane detection to improve the accuracy of lane detection. The Hough transform based on the straight lane lines do not work well for the curve lane lines [12].

El Houssein Chouaib Harik (2019) et al presented the work on A Novel LiDAR-Based Local Planner for Nonholonomic Mobile Robots. They presented in this paper a novel Heading Weight Function (HWF) as a local planner for a nonholonomic mobile robot. A LiDAR was used to detect the presence of obstacles in the robot's environment. The readings from the LiDAR were given as input to the HWF, which defined the best heading for the robot to avoid obstacles by weighing the LiDAR beam. Further enhanced by adding a LiDAR on the back of the robot or deploying a 3D LiDAR [].

Ayman M. Fouad (2019) et al presented the work on Fixed Obstacle Detection for Autonomous Vehicle. Camera and Raspberry pi used in the system and deep learning algorithm is used to detect the fixed obstacles in its pathway. Computer vision camera is not accurate in many atmospheric conditions [2].

S.O. Demir (2017) et al presented the work on An adaptive approach for road boundary detection using 2D LIDAR sensor. Experiments were conducted for both constructed and unconstructed roads, and the performance of the proposed method was evaluated. The accuracy results showed that using an adaptive approach for parameter estimation in the road detection algorithm. It is observed that the algorithm does not work fine for unstructured roads in which bumps and cracks on the road can prolong the line segment extraction and decision making processes [1].

R Ismail (2016) et al presented the work on Obstacle-avoiding robot with IR and PIR motion sensors. The developed robot will move in a particular direction once the infrared (IR) and the PIR passive infrared (PIR) sensors sense a signal

while avoiding the obstacles in its path. The system has lower accuracy, low flexibility and low precision [7].

Xuekui Zhu (2015) et al presented the work on A real-time Road Boundary Detection Algorithm Based on Driverless Cars. The extracted road boundary points are then straight line fitted using RANSAC algorithm. Finally, straight line is tracked using KALMAN filtering, then achieve real-time road boundary detection. Algorithm not working well in unstructured roads. The algorithm lacks vehicle information, such as the speed of vehicle, heading of vehicle and so on [16].

J. Han (2014) et al presented the work on Road boundary detection and tracking for structured and unstructured roads using a 2D lidar sensor. The algorithm extracts line segments from the roads; the endpoints of these line segments are then tracked using an NNF. The algorithm is inaccurate in different weather conditions [3].

O.Yalcin (2013) et al presented the work on Approaches of Road Boundary and Obstacle Detection Using LIDAR. The analysis of the approaches like positioning of Lidar, obstacle and boundary detection tested in VREP simulation environment. Performances and reliabilities of the algorithms is not efficiency [13].

Florian Janda (2013) et al presented the work on Road boundary detection for run-off road prevention based on the fusion of video and radar. Algorithm incorporates vision based road marking and road edge detection in the same algorithm. Both boundaries detected by the camera and the radar are represented by their own sets of parameters. Fusion of multiple sensors increases the cost and the algorithms detects the false edges [9].

2.2 PROBLEMS IN EXISTING METHODOLOGY

The existing methodologies for road boundary and obstacle detection in research area suffer from several problems, including:

- *Limited accuracy:* The accuracy of existing methods is often limited, particularly in challenging lighting and weather conditions, which can lead to false positives and false negatives.

- *High computational requirements:* Many existing methods require significant computational resources, which can make real-time implementation difficult, particularly on mobile devices.
- *Limited scalability:* Some existing methods are designed to work in specific environments and may not be easily scalable to different scenarios, such as different types of roads or different lighting conditions.
- *Limited generalization:* Many existing methods are designed to work with specific types of objects, such as cars or pedestrians, and may not be effective in detecting other types of obstacles, such as animals or debris.
- *Limited robustness:* Existing methods may not be robust to changes in the environment or to changes in the road surface, which can affect the accuracy of the detection.
- *Limited availability of annotated datasets:* Annotated datasets for road boundary and obstacle detection are limited in size and quality, which can hinder the development and evaluation of new methods.
- *Limited adaptability to new environments:* Existing methods may not be adaptable to new environments, such as new road constructions or urban planning changes, which can lead to poor performance in such scenarios.

CHAPTER 3

REQUIREMENT ANALYSIS

In this section, we discuss feasibility studies/risk analysis, software requirements & specification, hardware requirements and system use case of the obstacle detection & avoidance and road boundary detection system for autonomous vehicles.

3.1 HARDWARE REQUIREMENTS

In this project we are using RP LiDAR A2 M8, Raspberry Pi 4B for master communication, Arduino Mega for slave communication, Encoder Motors and Motor Drivers.

3.1.1 RP LiDAR A2 M8



Fig 3.1: RP LiDAR A2 M8

The RPLIDAR A2M8 is a 2D laser range scanner developed by RoboPeak, designed for accurate distance measurements in robotic applications. With a scanning frequency of 10Hz, it can measure distances up to 18 meters with a scanning range of 360 degrees. Using a Class 1 laser and a 2D motor for rotation, it generates up to 8000 samples per second. It uses a Time-of-Flight (ToF) system for distance measurements and is compatible with multiple operating systems and programming languages. Its low cost, ease of use, and reliability make it a suitable option for hobbyists and small-scale robotics projects.

Parameter	Values
Range	0.15 - 12 meters
Scan rate	5.5 - 10 Hz
Angular resolution	0.45 degrees
Sample rate	4000 samples/sec
Power consumption	2.5 W
Operating voltage	5V DC
Interface	USB or UART
Dimensions	102 mm x 72 mm x 84 mm
Weight	115 g

Table 3.1: LiDAR Parameters

Comparison of Lidar Sensor with various other sensors used in Road Boundary and Obstacle Detection using 2D LiDAR.

Parameter / Sensor	Ultrasonic	Radar	IR	Camera	LiDAR
Range	Short	Long	Short	Short	Medium
Range Accuracy	High	High	Poor	Medium	High
Angle Accuracy	None	Medium	None	High	High
Speed Measurement	No	Yes	No	No	No
Dust/Fog/Smoke/Robustness	Poor	High	Poor	Poor	Medium
Dark/Light Robustness	High	High	Medium	Medium	High
Integration into machines	Small	Small	Small	Medium	High
Cost	Low	High	Low	High	Very High

Table 3.2: Comparison of LiDAR with various sensors

3.1.2 25mm DC Metal Encoder Gearmotors



Fig 3.2: Encoder Motors

SKU: MOT3001-12V130RPM is a versatile and efficient gearmotor with a 12V DC source, built-in quadrature encoder, and metal gears. It has a no-load speed of 130 RPM with a gear ratio of 1:34, and a stall torque of 2.8 kg-cm (0.28 N-m) with a stall current of 1.2A. Its compact size, low current draw, and built-in encoder make it ideal for robotics, automation, and industrial control systems. With CE and RoHS compliance, it meets necessary safety and environmental standards, making it a reliable and safe option for a range of applications.

Parameter	Value
Operating Voltage	12V DC
No-load Speed	130 RPM
No-load Current	0.10A
Stall Torque	2.8 kg-cm (0.28 N-m)
Stall Current	1.2A
Gear Ratio	01:34
Encoder Resolution	11 CPR (counts per revolution)
Shaft Diameter	4mm
Motor Body Length	25mm
Operating Temperature	-10°C to 60°C

Table 3.3: Encoder Motor Parameters

3.1.3 Raspberry Pi 4B



Fig 3.3: Raspberry Pi 4B

The Raspberry Pi 4B is a versatile and powerful single-board computer designed for various applications, from basic desktop computing to industrial automation and robotics. It features a quad-core Cortex-A72 processor, up to 8GB of LPDDR4-3200 SDRAM, and excellent connectivity options, including Gigabit Ethernet, dual-band wireless, and Bluetooth 5.0. It also has various ports, such as USB 3.0, micro-HDMI, and a 40-pin GPIO header for easy interfacing with external devices. The Raspberry Pi 4B is a popular choice among hobbyists, developers, and industrial users because of its excellent hardware specifications and compatibility with various operating systems, including Raspberry Pi OS and Ubuntu.

3.1.4 Arduino Mega

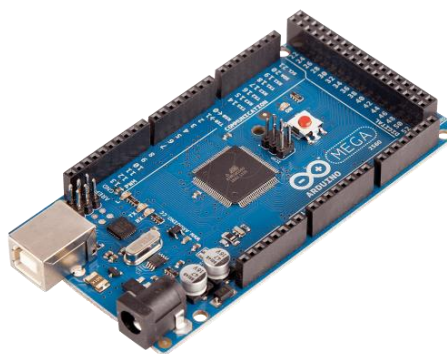


Fig 3.4: Arduino Mega

The Arduino Mega is a microcontroller board based on the ATmega2560 microcontroller, designed for more complex and larger projects. It has 256KB of flash memory, 8KB of SRAM, and 4KB of EEPROM memory, making it ideal for projects that require a lot of data storage or use of complex algorithms. It has 54 digital I/O pins and 16 analog input pins that can handle up to 20mA of DC current each. The board has a recommended input voltage of 7-12V and can be programmed using the Arduino IDE. With its high processing power and memory, the Arduino Mega is suitable for a wide range of applications, including robotics, automation, and data acquisition.

3.1.5 Motor Driver

The L298N motor driver is a popular dual H-bridge motor driver module used to control small to medium-sized DC motors. It has a voltage range of 5V to 35V and can deliver a peak current of 2A per channel and a continuous current of 1A per channel. It has heat sinks to prevent overheating during operation and can dissipate up to 25W of power.

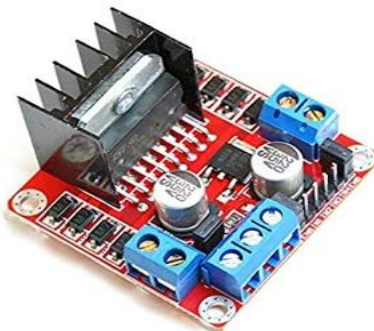


Fig 3.5: Motor Driver

The module is compatible with both DC motors and stepper motors, making it versatile, and has two channels for simultaneous control of two motors. Its dual H-bridge configuration allows for control of both speed and direction. The L298N motor driver is a reliable module for motor control systems used in robotics, automation, and other applications.

3.1.6 Batteries



Fig 3.6: Batteries

The 3.7V specification of a lithium-ion battery refers to its nominal voltage, which is the average voltage at which the battery operates. The 4.44Wh specification refers to the battery's energy capacity, which is the total amount of energy it can store and deliver. Lithium-ion batteries are popular for electronic devices due to their high energy density, long lifespan, good efficiency, and low self-discharge rates. The 3.7V 4.44Wh lithium-ion cell specification is valuable information for selecting and using the battery in electronic devices, ensuring efficient and effective operation.

3.1.7 Car Chassis

A car chassis for an autonomous robot is crucial to support its mechanical and electrical components. A good chassis should be sturdy, modular, and provide ample space for components.



Fig 3.7: Car Chassis

The weight should be optimized for the robot's size and intended use, and it should have an appropriate ground clearance. Customizability and reasonable cost are also essential. With the right chassis, the autonomous robot can perform tasks effectively, move smoothly, and interact with its environment.

3.2 SOFTWARE REQUIREMENTS

In this project we are using Arduino IDE, Raspbian OS, ROS Melodic, Matlab, RViz, Gazebo and Ubuntu 20.04.

3.2.1 Arduino IDE

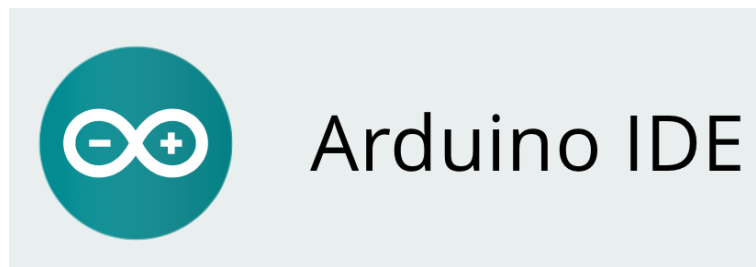


Fig 3.8: Arduino IDE

The Arduino Integrated Development Environment (IDE) is an open-source software application that simplifies programming and debugging of Arduino microcontroller boards. It uses the C++ programming language and includes a text editor, serial monitor, and pre-written code library. The IDE is compatible with a wide range of operating systems and Arduino boards, and provides tools for managing libraries, uploading code, and selecting the appropriate board and serial port. The Arduino IDE is easy to use and offers extensive documentation and tutorials, making it accessible to beginners and experienced programmers alike.

3.2.2 Raspbian OS

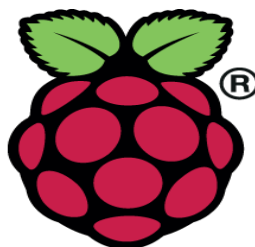


Fig 3.9: Raspbian OS

Raspbian is a free and open-source operating system that is specifically designed for the Raspberry Pi single-board computer. It is based on Debian and is optimized for the Raspberry Pi's ARM processor architecture, making it lightweight and fast. Raspbian has a user-friendly graphical interface and comes with a range of pre-installed applications, as well as access to a wide range of software through its package manager. It is regularly updated with security patches and new features, and can be run alongside other operating systems using a dual-boot setup. Raspbian is a popular choice among Raspberry Pi users for its ease of use and compatibility.

3.2.3 Ubuntu 20.04

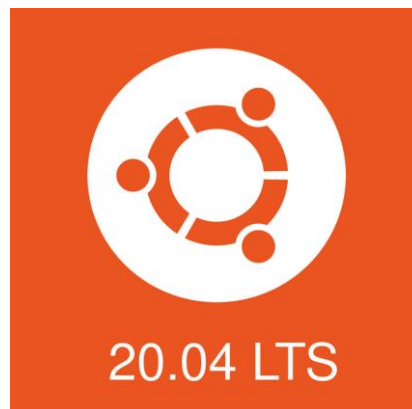


Fig 3.10: Ubuntu 20.04

Ubuntu 20.04 LTS is a free and open-source Linux operating system based on the Debian distribution. It offers a user-friendly interface, long-term support with security updates for five years, a Snap package manager for easy installation and management of software packages, improved performance, updated versions of popular software, strong security features, and built-in support for cloud services. Ubuntu 20.04 is a popular choice for both personal and enterprise use due to its modern features and reliability.

3.2.4 ROS Melodic



Fig 3.11: ROS Melodic

ROS Melodic is a flexible and scalable open-source middleware for robotics that offers a framework for developing robotic applications. Its distributed architecture enables nodes to communicate with each other using a publish-subscribe messaging pattern, allowing for the development of modular and scalable robotic applications. ROS Melodic includes a wide range of software packages that provide various functionalities for robotics, such as robot control, perception, and navigation, and it supports multiple programming languages. Additionally, ROS Melodic comes with development tools that simplify the development and testing of robotic applications. It supports several simulators, such as Gazebo and RViz, enabling developers to test their applications in a virtual environment before deploying them on actual robots. ROS Melodic has a large and active community of developers who provide support, tutorials, documentation, and forums to help new developers learn and use ROS. ROS Melodic is compatible with various operating systems and hardware platforms, making it accessible to a wide range of users.

3.2.5 RViz



Fig 3.12: RViz

RViz is a 3D visualization tool within ROS Melodic that enables users to visualize sensor data and robot models in a 3D environment, making it easier to debug robotic systems. It is highly configurable, allowing users to display and adjust the size and color of data from a wide range of sensors. RViz can display robot models, enabling users to visualize robot motion and check for collisions and errors. Its interactive tools allow users to manipulate objects directly and view detailed information. RViz is an essential tool for developers working with ROS Melodic, helping them debug and test robotics systems and validate algorithm and controller performance.

3.2.6 Gazebo



Fig 3.13: Gazebo

Gazebo is a 3D simulation environment that is a part of the Robot Operating System (ROS) Melodic distribution, widely used in the robotics industry and research community. It provides a physics engine that simulates the dynamics of objects in a 3D environment, allowing users to test and validate the performance of their robotic systems. Gazebo is highly customizable and can be extended with plugins to add new sensors, controllers, and other functionality. It is a valuable tool for robotics developers and researchers, as it allows them to test and validate the performance of their robotic systems in a simulated environment before deploying them in the real world.

3.3 RISK ANALYSIS OF 2D LIDAR

The use of 2D lidar systems has become increasingly popular in a wide range of applications, including autonomous vehicles, drones, and robotics. These systems use laser sensors to measure the distance to objects and create a 2D map of the surrounding environment. While these systems offer many benefits, such as

improved accuracy and reliability, they also present unique challenges and risks that must be addressed.

A risk analysis of using 2D lidar in both indoor and outdoor environments can help identify potential hazards and provide recommendations for mitigating those risks. In this article, we will explore in greater detail the specific risks associated with using 2D lidar in both environments.

3.3.1 Risk Analysis of 2D Lidar in Indoor Environment

- *Accuracy:* The 2D lidar system must be calibrated and configured for high accuracy to avoid hazardous incorrect detections.
- *Obstruction:* Proper placement of the lidar system is important to avoid obstacles like walls, pillars, or furniture that can cause incorrect detections and increase the risk of accidents.
- *Reflection and Interference:* Reflective surfaces and interference from other sensors can lead to incorrect distance measurement and false positives. Shielding the lidar system from such sources can reduce the risk of interference.
- *Integration:* Integration of 2D lidar with other systems can lead to incorrect data fusion and confusion between different sensors. To mitigate this risk, proper configuration and testing of the integration are required.
- *Malfunctions and failures:* To reduce the risk of incorrect detection and hazardous situations due to malfunctions and failures, redundancy and backup systems should be in place, and regular maintenance and testing of the system should be done.
- *Cybersecurity:* The lidar system may be vulnerable to cyber attacks that can compromise system integrity and lead to incorrect detections. Proper cybersecurity measures must be implemented to reduce this risk.

3.3.2 Risk Analysis of 2D Lidar in Outdoor Environment

- Environmental conditions can affect the performance of the 2D lidar system in an outdoor environment. The system must be configured and tested to operate under different weather conditions.

- Range is critical for outdoor applications, and the system must be capable of detecting objects at a safe distance with the required range and resolution.
- Obstruction can lead to incorrect detections, and the system should be placed in a location that provides a clear view of the surrounding environment.
- Interference from other sensors and systems, such as GPS signals, can lead to incorrect detections, and the system must be properly shielded.
- Power consumption can be challenging to manage in remote or off-grid locations, and power management systems should be in place to reduce the risk of power-related failures.
- Mechanical failure is a risk in harsh weather conditions, and the system should be designed and built to withstand environmental conditions, with regular maintenance and testing conducted to identify and mitigate potential failures.
- Cybersecurity measures should be in place to reduce the risk of cyber attacks on the system.

2D lidar systems offer benefits but present challenges and risks in indoor and outdoor environments. Risk analysis helps identify potential hazards and recommendations for mitigation. Proper calibration, configuration, and placement, as well as power management, environmental durability, and cybersecurity measures, are essential for high accuracy, safety, and performance.

3.4 FEASIBILITY STUDY OF 2D LIDAR OVER 3D LIDAR

In recent years, 2D lidar systems have become increasingly popular for various applications, including road boundary and obstacle detection. While 3D lidar systems are also available, there are several reasons why 2D lidar may be a more feasible option for this particular application.

- Cost-effective compared to 3D lidar, which is important for installations on multiple vehicles or roadside units

- Lower power consumption, making it a more feasible option for remote areas where power sources may be limited
- Higher scan rate, enabling quick and accurate detection of obstacles and road boundaries, particularly in high-speed applications like autonomous vehicles
- Compact and lightweight, making it easier to install and integrate into vehicles or roadside units, and reducing the load on the vehicle or unit for improved fuel efficiency and reduced emissions
- Wider field of view, making it easier to detect obstacles and road boundaries outside of the immediate area, even though 3D lidar provides more detailed information about the environment.

2D lidar is a cost-effective, low-power, high-scan-rate, compact, and wide field-of-view option for road boundary and obstacle detection, but the choice depends on specific requirements and feasibility analysis.

3.5 FEASIBILITY STUDY OF ENCODER MOTORS OVER NORMAL MOTORS

Autonomous cars require a robust and reliable propulsion system to safely navigate roads and highways. The propulsion system of an autonomous vehicle consists of an electric motor, a battery, and a controller. The motor is responsible for converting electrical energy into mechanical energy that propels the vehicle forward. Two types of motors commonly used in autonomous vehicles are encoder motors and normal motors.

- Encoder motors have a built-in sensor that tracks the motor's position and velocity, providing feedback to the controller for precise control. This makes them accurate and efficient, but they are more expensive and require additional wiring and hardware.
- Normal motors do not have a built-in sensor and rely on voltage and current to estimate speed and torque. They are less expensive and easier to integrate but can experience speed fluctuations, leading to inconsistent vehicle performance.
- Both motors can be used in autonomous vehicles, depending on the specific

application. Encoder motors are more suitable for high-performance applications that require precise control, while normal motors are better for low-speed applications.

- Cost is an important factor to consider, as encoder motors are more expensive, but the added cost may be justified for applications that require precise control. Power consumption is another factor, with encoder motors consuming less power and providing longer battery life and greater range for the vehicle.

Encoder motors provide precise control and low power consumption, but are more expensive and require additional wiring and hardware. Normal motors are less expensive and simpler to integrate, but can experience speed fluctuations. The choice depends on the specific application and requirements.

3.6 SYSTEM USE CASE OF THE PROJECT

Here are some more specific applications of road boundary and obstacle detection using 2D lidar:

- **Autonomous Vehicles:** As mentioned before, 2D lidar can play a crucial role in the development of autonomous vehicles. It can help detect the road boundaries, other vehicles, pedestrians, and other obstacles in real-time, enabling the vehicle to navigate safely and avoid collisions. With the help of 2D lidar, autonomous vehicles can adjust their speed and direction according to the road conditions and avoid unexpected obstacles.
- **Traffic Management:** The use of 2D lidar can also aid in traffic management. Traffic signals can be synchronized with the data generated from 2D lidar to avoid congestion and minimize the waiting time for vehicles at the intersections.
- **Urban Planning:** 2D lidar can be used to map the surrounding areas and provide a detailed analysis of the environment, including the road conditions, location of buildings, and other structures. This information can be used for urban planning, such as identifying areas where new roads need to be built or improving existing infrastructure.

- **Agricultural Robotics:** 2D lidar can also be used in agricultural robotics to detect obstacles, such as rocks or trees, that could damage the equipment. It can also help to map the terrain, providing information on the slopes and contours of the land, enabling the farmers to optimize their planting and harvesting processes.
- **Industrial Automation:** 2D lidar can be used in industrial automation to detect obstacles on the factory floor, such as pallets or boxes, allowing the robots to navigate around them safely. It can also be used to monitor the movement of goods and equipment, enabling the plant managers to optimize the production process.

The applications of 2D lidar for road boundary and obstacle detection are numerous, and its usage is not limited to just autonomous vehicles. Its benefits can be utilized in various industries, including agriculture, industrial automation, and urban planning, to name a few.

CHAPTER 4

PROPOSED METHODOLOGY

In this section, we discuss the overall functional architecture of the obstacle detection & avoidance and road boundary detection system for autonomous vehicles. The proposed architecture is shown, which will be deployed in autonomous vehicles. It consists of an obstacle detection module, obstacle avoidance module, and vehicle control system module, and are explained in the following subsections. Besides, the flow chart of the proposed obstacle detection and avoidance system is also explained.

4.1 FUNCTIONAL BLOCK OF PROPOSED METHODOLOGY

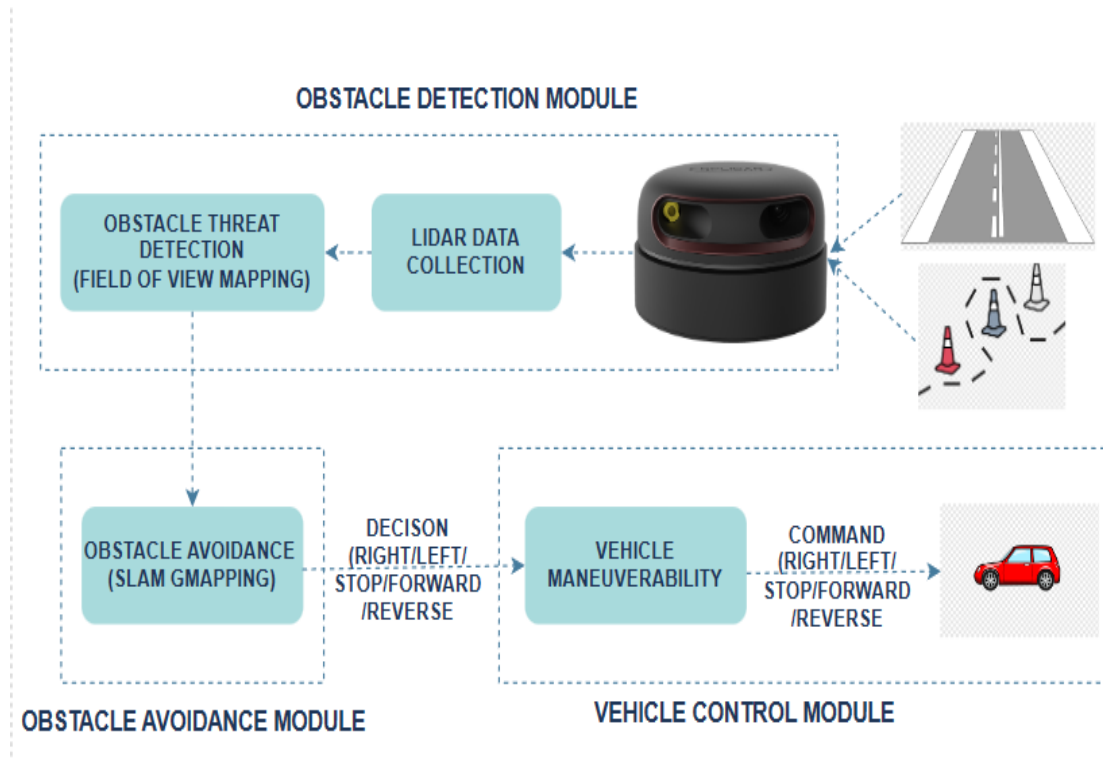


Fig:4.1: Block Diagram

1. Obstacle Detection Module

Autonomous vehicles work in a complicated environment full of obstacles such as people, buildings, and other vehicles. These barriers must be spotted by the autonomous vehicles in order for them to avoid collisions.

This section describes how the obstacle detection module recognises barriers in

front of autonomous ground vehicles. This module includes a lidar sensor module, lidar data collecting, lidar data mapping into the FOV, and an obstacle threat detection module, as defined below.

- *Lidar data collection*

Lidar data collection involves mounting a sensor on a moving platform to capture 3D point clouds that can be processed to extract features like building outlines and vegetation. Factors like the lidar sensor type, data point density, and environment can affect accuracy. Lidar data has various applications, including mapping, surveying, environmental monitoring, and autonomous vehicle navigation. It can be processed using software tools and machine learning algorithms to extract insights. Lidar data collection provides a highly detailed and accurate representation of the environment, making it an essential tool for many applications.

- *Lidar sensor assisted obstacle detection*

In general, autonomous vehicles get a hybrid vision from many modalities such as cameras, lidar, and radar. Lidar sensors perform reliably in a variety of settings when camera vision fails. As a result, the suggested lidar-based obstacle detection system may be used in autonomous ground vehicles to offer hybrid vision. To identify barriers in the operational environment, the obstacle detection module employs a lidar sensor. This module shoots and receives laser pulses in a controlled way on the autonomous vehicle's Field of Vision (FOV), where the orientation of the laser beams is known to the system.

2. Obstacle Avoidance Module

This module is in charge of making decisions to avoid obstacles. This module's services are only available when the obstacle detection layer detects the existence of barriers within critical distances. As a result, anytime the AGV senses an obstacle danger in its path, it simply contacts the obstacle avoidance module.

3. Vehicle Control System Module

This section describes the vehicle control system module, which is in charge of steering the autonomous vehicle. Via interrupt communication, this module gets directives from the main computer. The primary computer produces orders to steer the AGV by halting, turning left, or turning right based on the presence of impediments in the AGV's course.

4.2 FLOW DIAGRAM OF PROPOSED METHODOLOGY

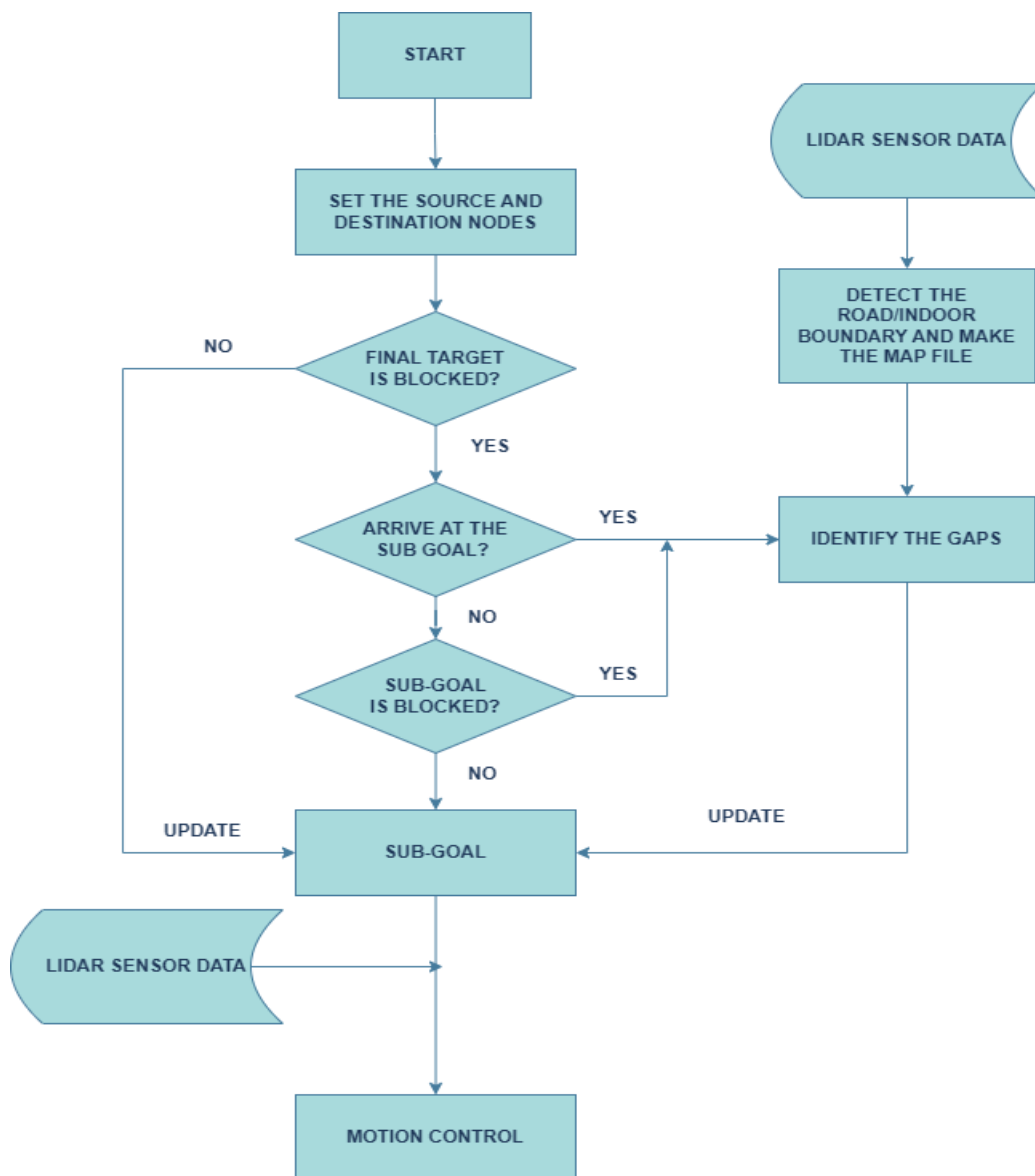


Fig 4.2: Flow Diagram

A major part of the proposed approach is identifying gaps of the environment based on 2D LIDAR . The method of identifying the gap is described in detail below. Finding a gap means that there is a large enough free space near the gap to allow the robot to pass. Then, choose one of the gaps we found as the new sub-goal, considering positions of the final target and previous sub-goals. According to the sub-goal and real-time sensor information, the robot's motion control is obtained. The specific method is described below. Compared with existing local algorithms that directly map the sensor data to the on-line control command, the proposed algorithm takes into account not only current sensor data but also other relevant information—the previous sub-goals. This is the key factor that keeps our algorithms from getting trapped. In every time interval, the navigation architecture is illustrated in Figure. The process iterates until the robot reaches the final goal.

4.3 COMMUNICATION MODEL OF PROPOSED METHODOLOGY

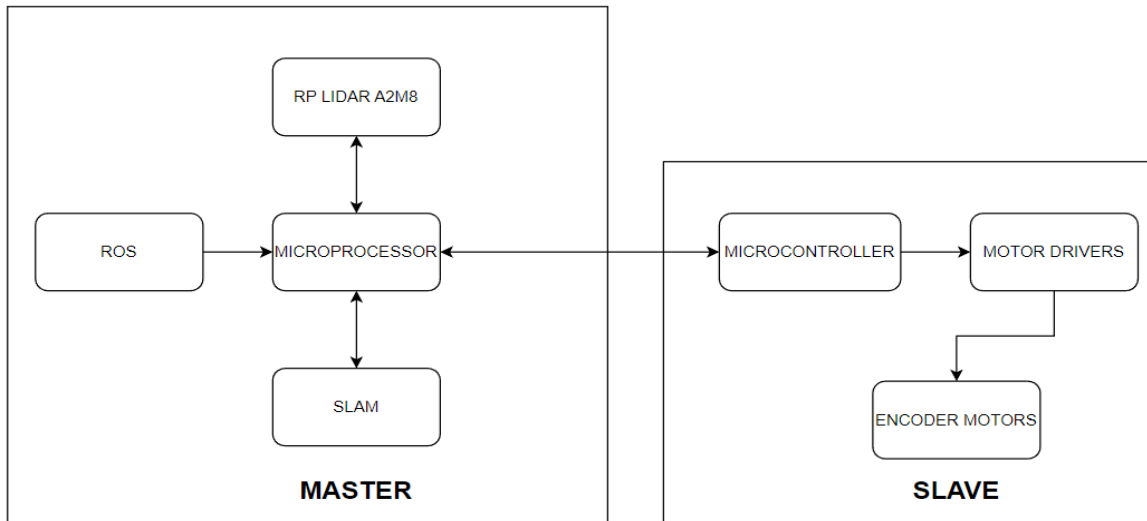


Fig 4.3: Communication Model

The proposed autonomous robot system employs Lidar sensor, SLAM algorithm, ROS operating system, Wi-Fi, RViz, Gazebo tools, slave device, encoder motors, and Matlab for obstacle avoidance and mapping. Lidar sensor is connected to a microprocessor loaded with ROS operating system, which is used for software development. The SLAM algorithm provides mapping and path planning for obstacle avoidance. RViz and Gazebo tools process and visualize the Lidar data. The motor control operations for the robot are performed by a slave device using encoder

motors for higher accuracy. The ROS toolbox provides an interface between Matlab and ROS for efficient data processing and path planning. The system uses a master-slave configuration technique for efficient communication between the microprocessor and the slave device, enabling precise control of the motor operations. The proposed system is a complex but impressive example of modern robotics technology, showcasing the potential for applications ranging from industrial automation to autonomous vehicles.

4.4 INTEGRATION OF ROS IN PROPOSED METHODOLOGY

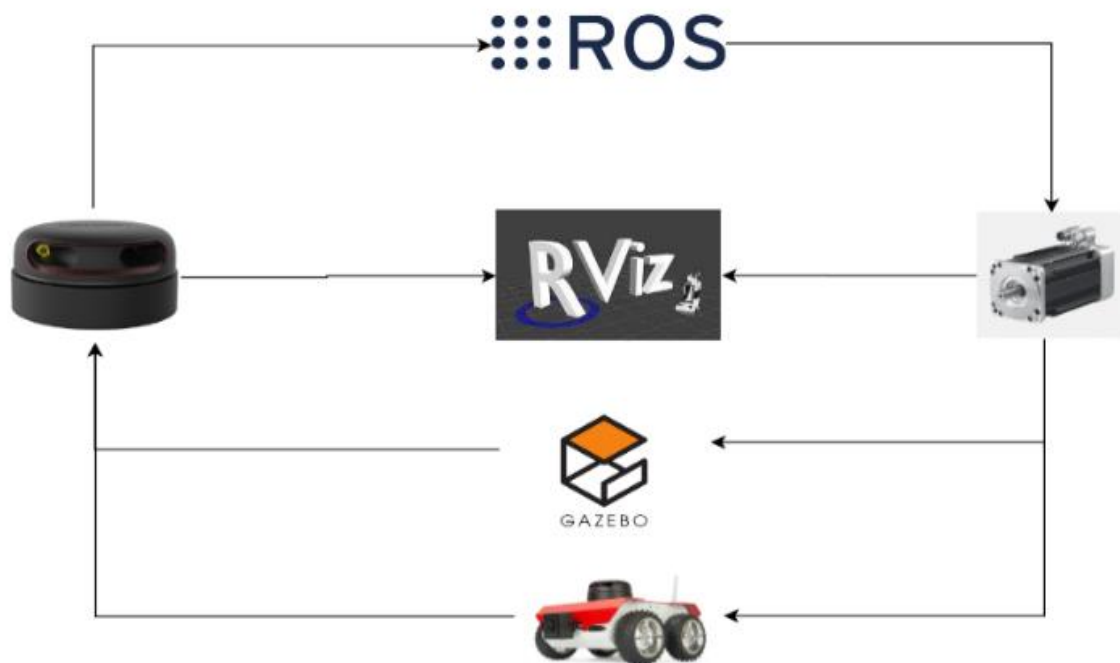


Fig 4.4: Integration of ROS

The integration of ROS, rviz, and Gazebo is a powerful tool for autonomous robots using 2D LIDAR sensors for localization and mapping in unknown environments. The use of 2D LIDAR sensors is popular due to their low cost and ability to provide a 360-degree view of the robot's surroundings. However, interpreting the generated data and processing it for mapping and localization algorithms can be challenging. ROS provides a common platform for integrating sensors, actuators, and software modules for performing tasks such as localization, mapping, and path planning. Rviz allows users to visualize sensor data and the robot's position and orientation in 3D space. Gazebo enables users to simulate the

behavior of a robot in a virtual environment, allowing them to test and evaluate algorithms in a range of scenarios without risking damage to expensive hardware. The integration of ROS, rviz, and Gazebo provides benefits such as real-time visualization and simulation, reduced testing time and cost, improved safety, and simplified development processes. It allows users to identify areas for improvement and refine algorithms for better performance. Additionally, it provides a common platform for integrating a wide range of components and algorithms, improving the overall performance of the robot.

4.5 SLAM TECHNIQUE

SLAM, or Simultaneous Localization and Mapping, is a fundamental technique for enabling autonomous robots and other systems to operate in unknown environments. The goal of SLAM is to enable a robot to build a map of its environment while simultaneously estimating its position within that map. This is done by using sensor data, such as laser range finders or cameras, to construct a model of the environment and the robot's position within it.

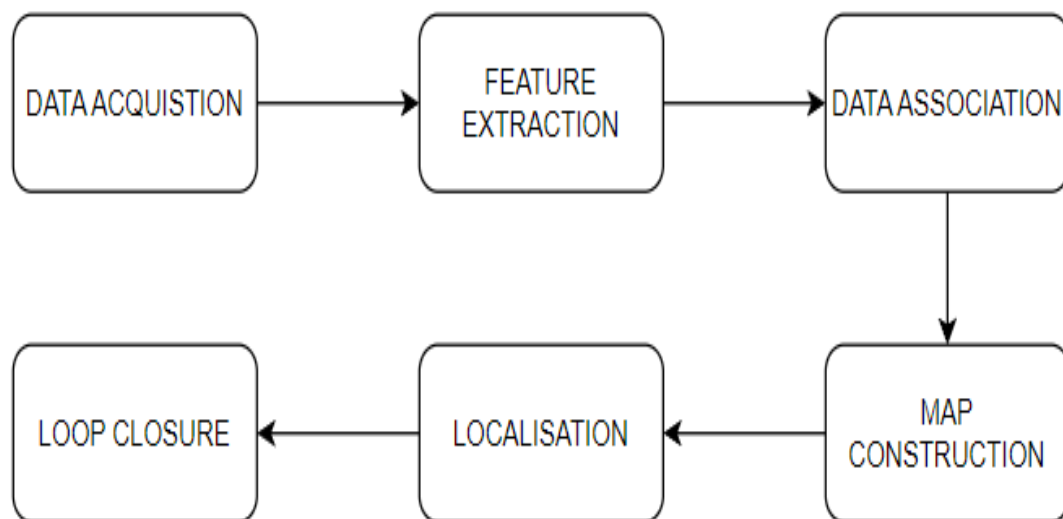


Fig 4.5: SLAM Technique

Simultaneous Localization and Mapping (SLAM) is a probabilistic algorithm that allows robots to estimate their position and map their environment using sensor data. The algorithm uses Bayesian inference to update the robot's belief of its position and the map as new sensor data is acquired, making it ideal for dealing with uncertainties in real-world environments. The SLAM algorithm consists of two

processes: mapping and localization, which are iteratively updated based on new sensor data. The data association problem is a key challenge in SLAM, where it can be difficult to match sensor measurements to features in the environment. Techniques such as feature extraction or clustering are used to identify unique features in the environment. Another challenge is the trade-off between accuracy and computational complexity, which is addressed by using sparse maps or reducing the dimensionality of the problem.

There are several SLAM algorithms available, each with its own strengths and weaknesses, including EKF-SLAM, FastSLAM, and GraphSLAM. SLAM has a range of applications, from self-driving cars to mobile robotics, allowing robots to navigate autonomously in unknown environments. Despite its challenges, SLAM is a powerful tool that has enabled significant advances in autonomous systems. As sensor technology improves and computing power becomes more affordable, SLAM is expected to continue playing an increasingly important role in enabling autonomous systems to operate in various applications.

4.6 SLAM GMAPPING ALGORITHM

SLAM (Simultaneous Localization and Mapping) is a process of building a map of an unknown environment while simultaneously localizing a robot within that environment. This process is essential for robots to navigate in unknown or changing environments without relying on pre-existing maps. One of the most popular SLAM algorithms is the gmapping algorithm.

The gmapping algorithm is a probabilistic approach that uses sensor data from a robot to create a map of the environment. It is a grid-based mapping algorithm that utilizes a probabilistic occupancy grid representation of the environment. The algorithm builds the map by estimating the probability of occupancy of each cell in the grid based on sensor measurements and robot poses. The robot pose refers to the robot's position and orientation in the environment.

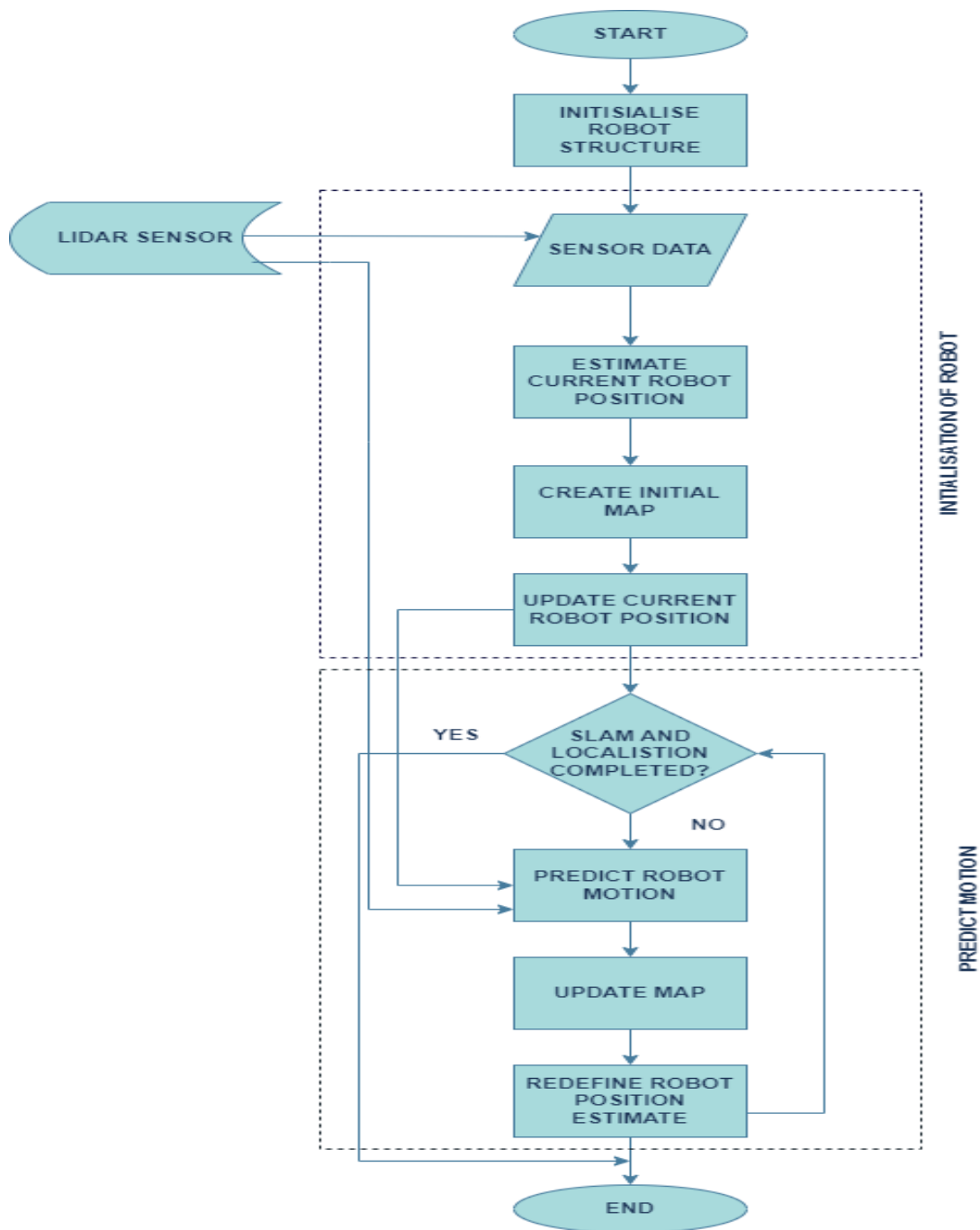


Fig 4.6 SLAM Gmapping Flow Chart

The gmapping algorithm has two main components: the motion model and the sensor model. The motion model is used to estimate the robot's pose based on the odometry data (e.g., wheel encoders). The sensor model estimates the probability of occupancy of each cell in the grid map based on the sensor data (e.g., laser range finder or sonar).

The algorithm works as follows:

- Initialize the occupancy grid map with uniform probabilities for all cells.
- Receive sensor data and robot odometry data.
- Use the motion model to estimate the robot's pose at the current time step.
- Use the sensor model to update the probabilities of occupancy for each cell in the grid map.
- Repeat steps 2-4 for each time step.
- Output the final grid map with probabilities of occupancy for each cell.

The mathematical representation of the gmapping algorithm can be expressed as follows:

- Let $M = \{m_1, m_2, \dots, m_n\}$ be the occupancy grid map, where each m_i represents the probability of occupancy of cell i .
- Let $Z = \{z_1, z_2, \dots, z_k\}$ be the sensor measurements, where each z_i is a range measurement from the robot's sensors.
- Let $U = \{u_1, u_2, \dots, u_t\}$ be the robot odometry data, where each u_i represents the robot's movement at time t .

4.7 PROJECT MANAGEMENT PLAN

This project has been carried out in two phases of work in Semester 7 and Semester 8 respectively.

In Phase 1 of the work, we have done detailed research over the project domain, defining the title, conducting a literature review over the defined title, and proposing the novel idea of the work.

In Phase 2 of the work, we defined the proposed project block diagram, the flow of the project, the best algorithm for the proposed project, the suitable software, stimulated the proposed project and real-time implementation, and finally published the research paper at the international conference.

PROJECT PHASE – I MANAGEMENT	
Month	Work
July	Defining project domain
August	Defining Project Title
September	Literature of the Project Title
Ocotober	Abstract and Rough work on Proposed Methodology
November	Research Paper Work and Project Phase I report work

Table 4.1: Project Phase -I Management

PROJECT PHASE – II MANAGEMENT	
Month	Work
January	Defining Proposed Methodology and Flow of the Project
February	Implementation of Proposed Methodology in stimulation environment
March	Implementation of Proposed Methodology in real-time environment
April	Testing and Report work for Project Phase II and publishing Research Paper

Table 4.2: Project Phase -II Management

4.8 COST ESTIMATION OF THE PROJECT

Estimating the cost of an autonomous car for road boundary and obstacle detection using 2D lidar will depend on various factors, such as the quality and quantity of the components, and the cost of labour if any. Here's a breakdown of the components you mentioned and their estimated costs:

COST ESTIMATION	
Component	Price
Motor Driver	200
Screws and Connecting Wire	500
Batteries	1,000
Arduino Mega	1,600
Car Chassis	2,000
Encoder Motors	2,600
Raspberry Pi 4B	14,500
RP LiDAR A2 M8	25,700
TOTAL	48,100

Table 4.3: Cost Estimation

CHAPTER 5

PROJECT IMPLEMENTATION

This chapter address the comprehensive implementation of the proposed methodology with the processes involved in the implementation of the suggested methodology. Then we describe the algorithm implementation and testing of the algorithm.

5.1 ROBOT DESIGN

We have built a autonomous rover robot with the 3 layer stack for placing the required components in the every stack layer.

Stack Layers of the Robot

The proposed system consists of three layers, each with a specific function to enable the robot's mechanical and electrical functionality.

Bottom Layer: The bottom layer comprises an Arduino Mega board, which is responsible for controlling the encoder actuators. The encoder motors are connected to motor drivers, which enable them to receive the required power supply to function optimally. The Arduino Mega board's primary role is to receive and process input signals from the Raspberry Pi board and convert them into output signals that will control the motor drivers. This layer's proper functioning is critical to the robot's overall performance, and any failure could result in the robot's mechanical malfunction.

Middle Layer: The middle layer comprises the Raspberry Pi, which is the core of the system responsible for controlling the entire robot's operation. This layer also includes a power bank that provides the necessary power supply for the Raspberry Pi board. The Raspberry Pi board receives input signals from the Lidar sensor located at the upper layer of the robot. The board then processes these signals and sends output signals to the Arduino Mega board to control the encoder motors' movements. This layer's proper functioning is essential to the robot's overall performance and requires constant monitoring to detect and fix any errors.

Upper Layer: The upper layer includes the Lidar sensor, which is the primary sensor used to detect road boundaries and obstacles. This layer also contains batteries that provide the required power supply for the encoder motors. The Lidar sensor is mounted at the top of the robot and has a 360-degree view, enabling it to detect any obstacles in the robot's path. The sensor's output signals are transmitted to the Raspberry Pi board for processing and analysis. This layer's proper functioning is crucial to the robot's overall performance, and any malfunction could result in the robot colliding with obstacles or deviating from the set path.

The mechanical functioning of the robot is achieved through the connection of encoder motors, wheels, and castor wheels at the base of the stack layers. The encoder motors are responsible for moving the robot's wheels, while the castor wheels provide support and balance to the robot. The proper functioning of the mechanical components is critical to the robot's overall performance and requires regular maintenance to ensure optimal performance.

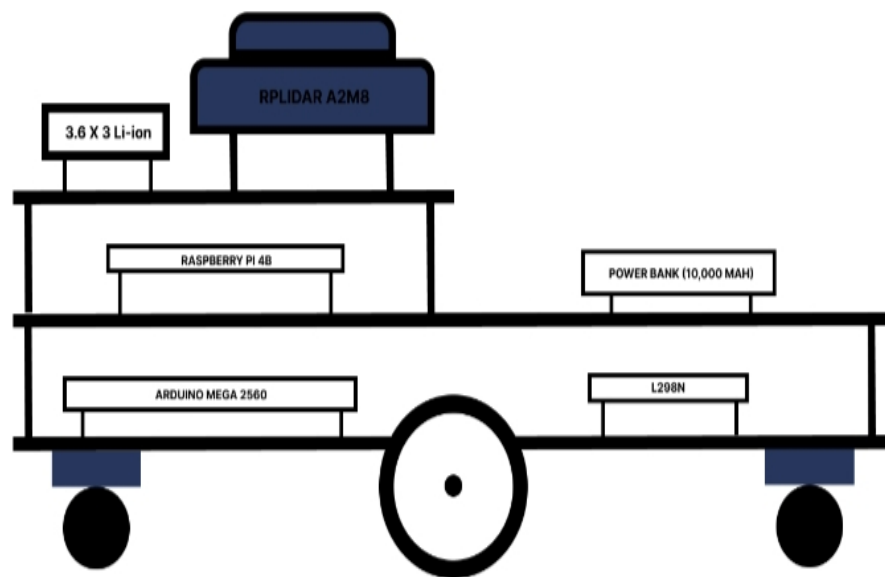


Fig 5.1: Robot Setup

5.2 RP LiDAR A2 M8 TESTING

Next step is to clone the RP LiDAR ros library package for integration with the Raspberry Pi 4B. The following Linux terminal commands for cloning the package, compiling the packages and running the LiDAR is detailed below,

\$ git clone https://github.com/Slamtec/rplidar_ros.git

'git clone https://github.com/Slamtec/rplidar_ros.git' is a command used to clone the 'rplidar_ros' package from the GitHub website. This package is a ROS driver that enables communication between ROS and the RPLIDAR laser scanner, allowing you to use the lidar data in your own ROS projects.

\$ catkin_make

'catkin_make' is a command used in ROS to build packages and workspaces by compiling and linking source code. It is an essential step in the software development process, as it ensures that changes to the codebase are incorporated correctly.

\$ roslaunch rplidar_ros rplidar.launch

'roslaunch rplidar_ros rplidar.launch' is a terminal command used to launch the RPLIDAR ROS package and start the ROS nodes required to communicate with the RPLIDAR laser scanner. This command is a crucial step in using the RPLIDAR laser scanner with ROS and enables us to visualize and analyze the lidar data captured by the sensor.

\$ rviz

'rviz' command is used to launch Rviz, a powerful 3D visualization tool for ROS. With Rviz, you can visualize and analyze data from sensors and robots in real-time, making it a valuable tool for robotics developers and researchers.

5.3 COMMUNICATION BETWEEN MASTER AND SLAVE

ROS Serial is a ROS package that enables serial communication between a ROS node running on a computer and a microcontroller board such as an Arduino. With ROS Serial, you can easily send and receive data between your ROS environment and your microcontroller board. To use ROS Serial to communicate between an Arduino Mega and a Raspberry Pi, you need to follow the following steps:

- Install ROS Serial on both the Raspberry Pi and the Arduino Mega.
- Create a ROS node on the Raspberry Pi that subscribes to a ROS topic and

sends data to the Arduino Mega over serial.

- Create an Arduino sketch that reads data from the serial port and performs the necessary actions based on the received data.
- Connect the Raspberry Pi and the Arduino Mega using a USB cable.
- Launch the ROS Serial node on the Raspberry Pi and the Arduino Mega.
- Launch the ROS node on the Raspberry Pi that sends data to the Arduino Mega.
- Verify that the Arduino Mega is receiving data and performing the expected actions.

By following these steps, you can establish communication between an Arduino Mega and a Raspberry Pi using ROS Serial. This can be useful in a variety of robotics applications, such as controlling motors or sensors connected to the Arduino from the Raspberry Pi running ROS. The following Linux terminal commands has been install ROS serial and establish ROS serial

\$ git clone https://github.com/Slamtec/rplidar_ros.git

It is a terminal command used to clone the 'rplidar_ros' package from the GitHub repository maintained by Slamtec. This package is a ROS driver that enables communication between ROS and the RPLIDAR laser scanner, allowing you to use the lidar data in your own ROS projects.

\$ roscore

'roscore' terminal command starts the ROS Master process and initializes the ROS environment. Running 'roscore' is typically the first step in any ROS project, and it provides a centralized communication hub for ROS nodes to discover and communicate with each other.

\$ rosrun roserial_python serial_node.py /dev/ttyACM0

'rosrun roserial_python serial_node.py /dev/ttyACM0' command is used to run the 'serial_node.py' script in the 'roserial_python' ROS package, which enables serial communication between a computer running ROS and an Arduino microcontroller over the specified serial port.

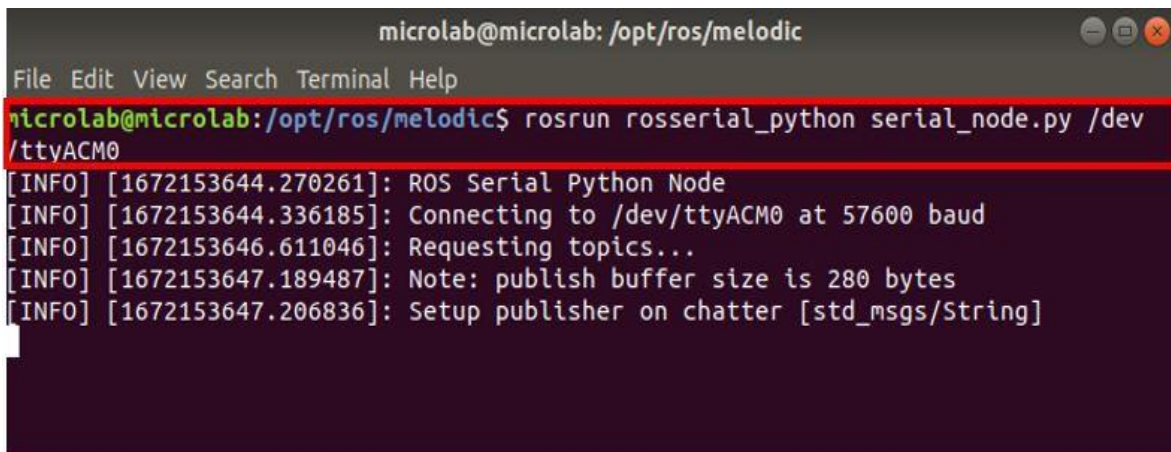
A screenshot of a Linux terminal window. The title bar at the top reads "microlab@microlab: /opt/ros/melodic". Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal prompt is "microlab@microlab:/opt/ros/melodic\$". The command entered is "roslaunch rosserial_python serial_node.py /dev/ttyACM0", which is highlighted with a red rectangular box. Below the command, several lines of output are displayed in a monospaced font: "[INFO] [1672153644.270261]: ROS Serial Python Node", "[INFO] [1672153644.336185]: Connecting to /dev/ttyACM0 at 57600 baud", "[INFO] [1672153646.611046]: Requesting topics...", "[INFO] [1672153647.189487]: Note: publish buffer size is 280 bytes", and "[INFO] [1672153647.206836]: Setup publisher on chatter [std_msgs/String]". The terminal background is dark purple.

Fig 5.2: ROS Serial Master Slave

5.4 TELEOP OF THE ROBOT

The teleop twist keyboard is a ROS package that provides a simple way to control a ROS robot using the keyboard. It allows you to send velocity commands to a robot using the arrow keys on your keyboard.

Arduino has been developed according to the required functionalities for the project, and it has been included in an appendix for reference. This is an important step towards building a fully functional robot that can respond to commands from the Raspberry Pi. The following Linux Terminal commands have been used for Teleop of the robot.

\$ git clone https://github.com/ros-teleop/teleop_twist_keyboard.git

The terminal command "git clone https://github.com/ros-teleop/teleop_twist_keyboard.git" is used to clone the code repository for the teleop twist keyboard package. This package provides a way to control a robot using keyboard input from a computer, and is often used for testing and development purposes.

\$ catkin_make

'catkin_make' is a command used in ROS to build packages and workspaces by compiling and linking source code. It is an essential step in the software development process, as it ensures that changes to the codebase are incorporated correctly.

\$ roscore

'roscore' terminal command starts the ROS Master process and initializes the ROS environment. Running 'roscore' is typically the first step in any ROS project, and it provides a centralized communication hub for ROS nodes to discover and communicate with each other.

\$ 'roslaunch teleop_twist_keyboard teleop_twist_keyboard.py'

The terminal command "roslaunch teleop_twist_keyboard teleop_twist_keyboard.py" is used to launch the teleop twist keyboard node, which allows you to control a robot's movement using keyboard input from a computer.

Now that the teleop twist keyboard package is installed and the node is running, the robot can be controlled using keyboard inputs.

5.5 TICK VALUE MEASUREMENT

The tick value per meter for a quadrature encoder is a measure of how many pulses or "ticks" the encoder generates for each unit of linear distance traveled by the device that the encoder is attached to. This value is determined by the resolution of the encoder, the gear ratio (if applicable), and the circumference of the wheel or pulley that the encoder is attached to it. To calculate the tick value per meter, you can use the following formula:

Tick value per meter = (Encoder resolution * Gear ratio) / Circumference of wheel

Theoretical Tick Value Measurement:

We have an encoder with a resolution of 11 pulses per revolution, and also with the gear ratio 1:46 and it is attached to a wheel with a circumference of 0.2198 meters, the tick value per meter would be:

Tick Value per meter = $(11 \times 46) / 0.2198 = 2302$

Practical Tick Value Measurement:

We have conducted the trial-and-error method for the calculating the tick value of the robot. We moved the robot the 5 times manually and noted the tick value for each step. Average of the tick value calculated at each step gives the final tick value.

S.No	Tick Value
1	2562
2	2562
3	2525
4	2575
5	2567
Average	2558.2

Table 5.1: Tick Value Observation

Percentage of Error:

To calculate the percentage error between the theoretical and practical tick value measurements, we can use the following formula:

$$\text{Percentage error} = |(\text{Theoretical value} - \text{Practical value}) / \text{Theoretical value}| * 100\%$$

Substituting the values given in the question, we get:

$$\text{Percentage error} = |(2302 - 2558.2) / 2302| * 100\% = 10.0\%$$

Therefore, the percentage error between the theoretical and practical tick value measurements is 10.0%. This means that the practical measurement differs from the theoretical measurement by 10%, which is within a reasonable range of error for this type of measurement.

The calculated practical tick value is taken as the parameter for the differential drive package. The differential drive package provides a convenient and standardized way to control differential drive robots, and can greatly simplify the process of developing and testing robot control algorithms. The following Linux Terminal command is to clone and install the package in ROS.

\$ git clone <https://github.com/jfstepha/differential-drive.git>

The command "git clone" is a command used in the Linux terminal to make a copy of a repository from GitHub onto your local computer. In this specific example, it is being used to clone the repository located at <https://github.com/jfstepha/differential-drive.git>.

\$ catkin_make

'catkin_make' is a command used in ROS to build packages and workspaces by compiling and linking source code. It is an essential step in the software development process, as it ensures that changes to the codebase are incorporated correctly.

5.6 URDF GENERATION

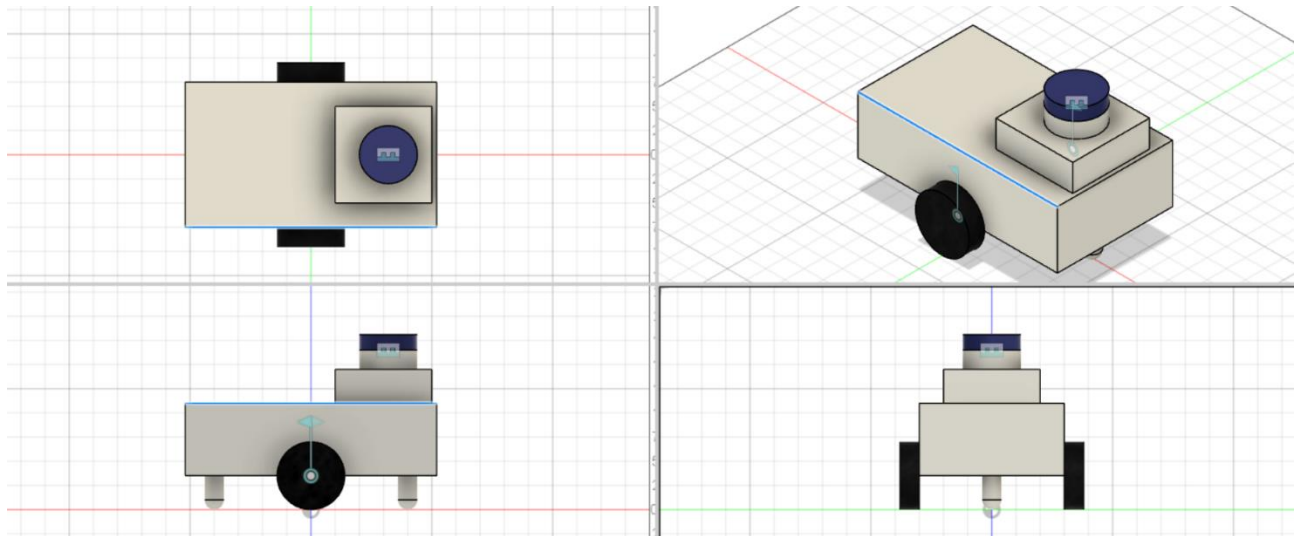


Fig 5.3: URDF of the Robot

URDF (Unified Robot Description Format) is an XML file format used to describe the physical properties and geometry of a robot in a standardized way. In this case, we will be generating a URDF for a mobile robot with RP Lidar, Raspberry Pi, Arduino Mega, car chassis, wheels, and castor wheels.

The first step in generating a URDF for the mobile robot is to define its geometry. This can be done using a 3D modeling software, such as SolidWorks or Blender, which allows you to create a detailed model of the robot's physical components. Once the geometry is defined, it can be exported in a format that can be used by the URDF.

Next, we need to define the robot's joints and links. A joint represents a movable connection between two links, while a link represents a rigid body in the robot. In our case, we have several links, including the car chassis, wheels, and castor wheels, which are connected by joints. To define the joints and links, we use the URDF syntax to specify their names, types, and properties. For example,

we can define a revolute joint for the wheels, which allows them to rotate around a single axis, and specify the maximum and minimum angles of rotation. We can also specify the dimensions and mass properties of each link, as well as its visual and collision properties. Once the joints and links are defined, we can add sensors to the robot, such as the RP Lidar. To do this, we can use the URDF syntax to specify the sensor's name, type, and position relative to the robot's links. We can also specify the sensor's properties, such as its field of view and resolution.

In addition to the physical components of the robot, we also need to define its controllers and actuators. The Raspberry Pi and Arduino Mega can be used as the controllers for the robot, while the wheels and castor wheels can be controlled using motors and servos. We can use the URDF syntax to specify the controller and actuator names, types, and properties, as well as their connections to the robot's joints.

Finally, we can combine all of the components of the URDF into a single file, which can be loaded into simulation environments, such as Gazebo or RViz, to visualize and test the robot's behavior. The URDF file can also be used to generate control code for the robot, such as ROS (Robot Operating System) nodes, which can be used to control the robot in real-world applications.

5.7 MAPPING THE ENVIRONMENT

The `slam_gmapping` package is a ROS (Robot Operating System) package that provides a node for performing Simultaneous Localization and Mapping (SLAM) using a laser range finder sensor (such as a LIDAR). Here's a brief overview of the key components of the `slam_gmapping` package:

- *slam_gmapping node*: This is the main node of the package that performs SLAM using the laser range finder sensor data. It uses the Grid-based FastSLAM algorithm to perform SLAM, which maintains a map of the environment as well as the robot's pose within that environment.
- *gmapping library*: This is the core library that provides the implementation of the Grid-based FastSLAM algorithm. It is responsible for maintaining the map and the robot's pose estimates, as well as performing updates based on incoming sensor data.

- *scanmatcher library*: This library provides the implementation of the scan matching algorithm, which is used to align the laser scan data with the map to estimate the robot's pose.
- *tf package*: This is a ROS package that provides tools for working with coordinate frames. The `slam_gmapping` node uses the `tf` package to maintain the coordinate frames of the robot and the laser range finder sensor.

The `slam_gmapping` package is commonly used in ROS-based robotics projects for mapping the environment and localizing the robot within that environment using laser range finder data. It can be configured to work with different types of laser range finders and can be integrated with other ROS nodes to provide a complete robotics solution. Here the terminal commands used for running the mapping the environment.

\$ git clone https://github.com/ros-perception/slam_gmapping.git

The command `git clone https://github.com/ros-perception/ slam_gmapping.git` is used to clone the `slam_gmapping` repository from GitHub onto your local machine.

\$ catkin_make

'`catkin_make`' is a command used in ROS to build packages and workspaces by compiling and linking source code. It is an essential step in the software development process, as it ensures that changes to the codebase are incorporated correctly.

\$ roslaunch sanlog_description display.launch

The command `roslaunch sanlog_description display.launch` launches a ROS launch file called `display.launch` in the `sanlog_description` package. This launch file is typically used to visualize a robot model in RViz, a 3D visualization tool for ROS. The launch file may include a `robot_state_publisher` node to publish the transform between the robot's base link and all of its joints, and a `rviz` node to

launch the RViz visualization tool with the appropriate configuration to display the robot model and sensor data.

\$ roslaunch slam_gmapping gmapping scan:=/scan

The command `roslaunch slam_gmapping gmapping scan:=/scan` launches the gmapping node from the `slam_gmapping` package in ROS, which performs Simultaneous Localization and Mapping (SLAM) using laser range finder sensor data. This command is used to launch the gmapping node with the appropriate topic configuration for laser range finder data. It overrides the default behavior of the gmapping node, which subscribes to the `/base_scan` topic, and instead subscribes to the `/scan` topic to receive laser range finder data.

\$ rosrun map_server map_saver -f map

The `roslaunch map_server map_saver -f map` command in ROS is used to save a map generated by a mapping algorithm to a file on your local system. It runs the `map_saver` node from the `map_server` package, and the output file is specified using the `-f` option followed by the desired file name, which in this case is `map`.

5.8 NAVIGATION OF THE ROBOT

GMapping is a popular open-source SLAM (Simultaneous Localization and Mapping) algorithm that is often used in mobile robots for mapping and navigation. It is integrated with the ROS (Robot Operating System) framework and provides real-time 2D mapping capability by utilizing sensor data, such as laser range finders, to construct a map of the environment. The GMapping algorithm uses a particle filter-based approach to estimate the robot's pose and the map of the environment. The algorithm starts with a uniform distribution of particles, each representing a possible location and orientation of the robot in the environment. As the robot moves and collects sensor data, the particles are updated and resampled, resulting in a new set of particles that more accurately represent the robot's pose. The algorithm also updates the map based on the sensor data and the estimated robot pose. Once the map is generated, it can be used for navigation by the robot. ROS provides a navigation stack that includes path planning and obstacle avoidance algorithms, which can be used with the GMapping-generated

map to plan a path and guide the robot to a desired location. The following Linux Terminal command used for navigating the robot.

\$ roslaunch amr_navigation amr_navigation_final_launch

The `roslaunch amr_navigation amr_navigation_final_launch` Linux command is used to launch the AMR (Autonomous Mobile Robot) navigation system, which is a part of the ROS (Robot Operating System) framework. This command executes the `amr_navigation_final_launch` launch file, which contains the configuration and startup instructions for the navigation system, including parameters for the robot's sensors and the navigation stack. The behavior of the navigation system is determined by the specific configuration and parameters set in the launch file.

CHAPTER 6

RESULT AND DISCUSSION

This chapter elaborates the stimulation results analysis and real time environment results. The stimulation results and analysis provide valuable insights into the system's performance in a controlled environment. On the other hand, the real-time environment results demonstrate how the system performs under real-world conditions. Both sets of results are essential in evaluating the system's effectiveness and identifying areas that require improvement.

6.1 STIMULATION RESULTS

The proposed system was implemented in a simulation environment using ROS operating with RViz and Gazebo tools. This environment allowed for testing the system's performance in a controlled setting before deploying it in a real-world scenario. Additionally, the Lidar data collected in ROS was integrated with MATLAB for efficient processing and analysis, providing valuable insights into the data's characteristics and patterns.



Fig 6.1: Indoor Environment

Figure 6.1 depicts the indoor environment in the Gazebo toolbox; further, the indoor environment is mapped by 2D Lidar using the RViz toolbox. The operations for mapping the environment are performed by node location over Wi-Fi in a real-time environment.

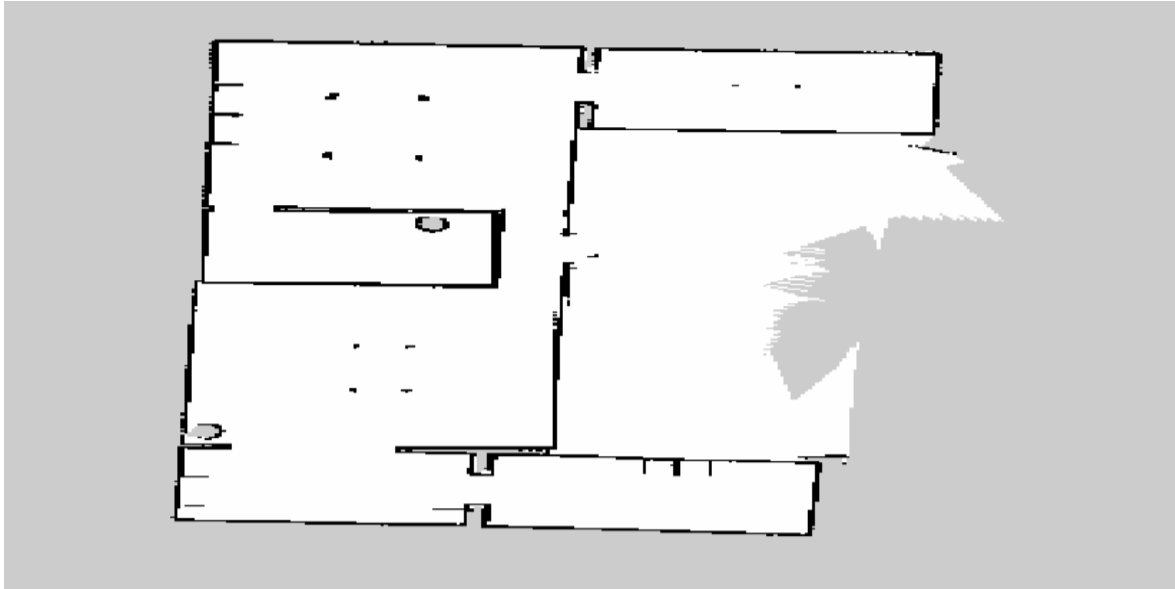


Fig 6.2: Mapped environment

Figure 6.2 depicts the mapped indoor environment visualised in RViz. The processed map data of 2D Lidar is stored as a YAML file for path planning and obstacle avoidance in the Gazebo environment.

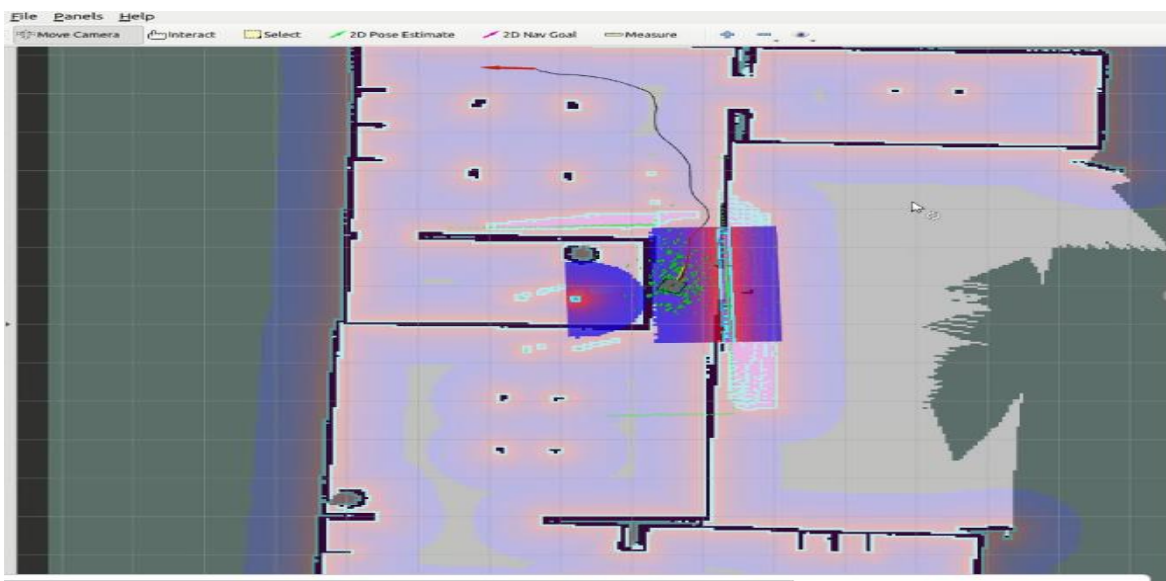


Fig 6.3: Robot mapped to destination with avoiding obstacles

Further, the node point of the destination location is pointed in the Rviz stimulation tool, and then the robot finds the best possible path for reaching the destination point while avoiding obstacles.

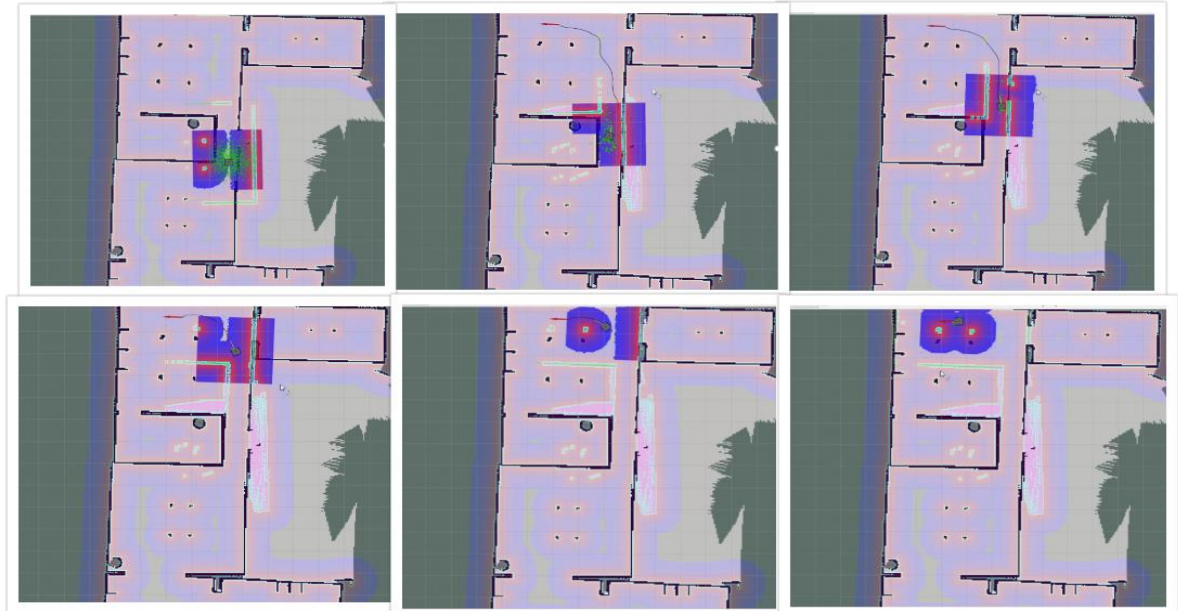


Fig 6.4: Path planning and obstacle avoidance of the robot

Figure 6.4 depicts how the robot moves to the destination point with path planning and obstacle avoidance techniques. Similarly, the real-time environment stimulated and visualisation for path planning and obstacles can be viewed in the gazebo toolbox in figure 6.5.

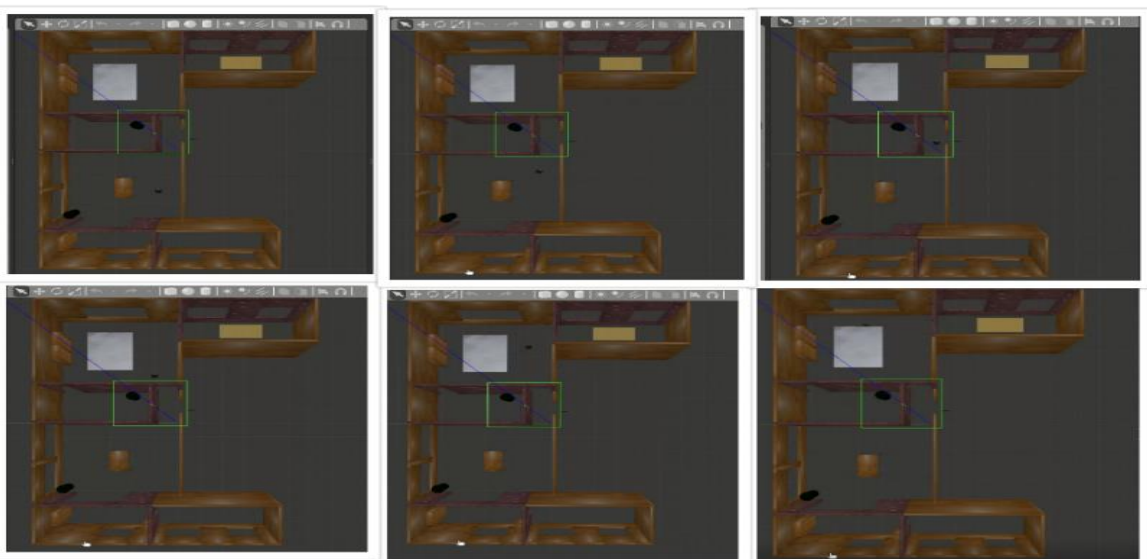


Fig 6.5: Path planning and obstacle avoidance stimulation in gazebo tool box

6.2 REAL TIME ENVIRONMENT RESULT

This subsection detailed the results output in the real time environment. Figure 6.6 depicts the straight view of the robot model and Figure 6.7 depicts the side view of the robot model.

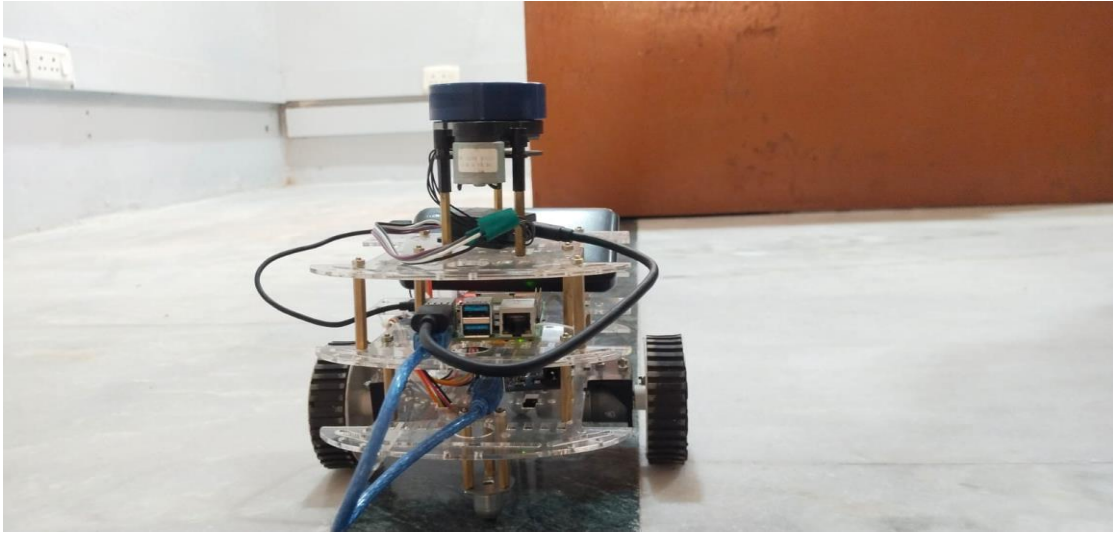


Fig 6.6: Straight view of the Robot

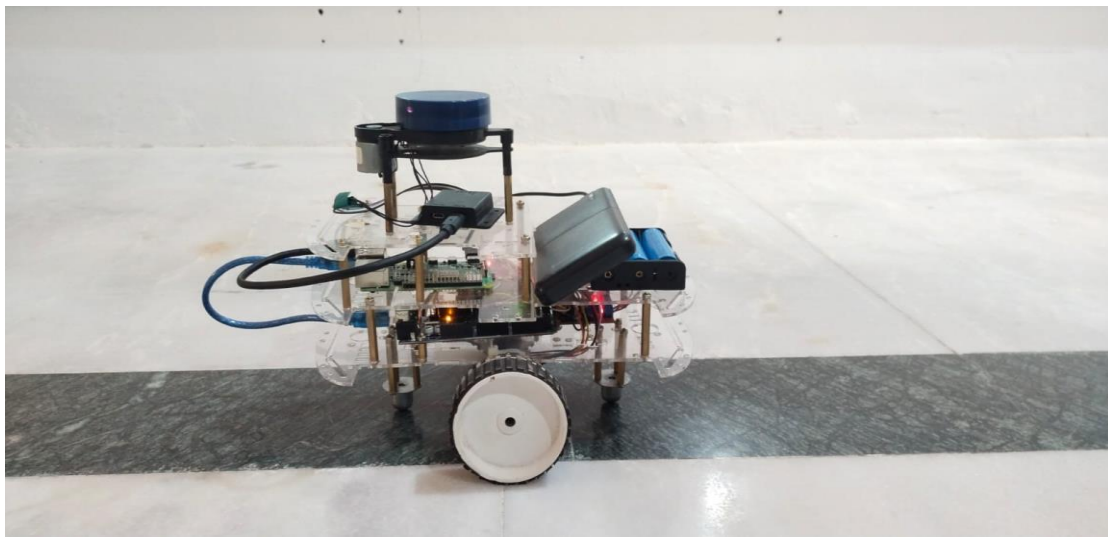


Fig 6.7: Side view of the Robot

After placing the robot in the real time environment, the robot needs to map the real time environment. Figure 6.8 depicts the real time environment to be mapped for the robot.



Fig 6.8: Real time environment

After placing the robot in the real time environment, the 2D LiDAR and mapping should be started using the Linux Terminal commands mentioned in Chapter 5. Figure 6.9 depicts the mapping of the environment.

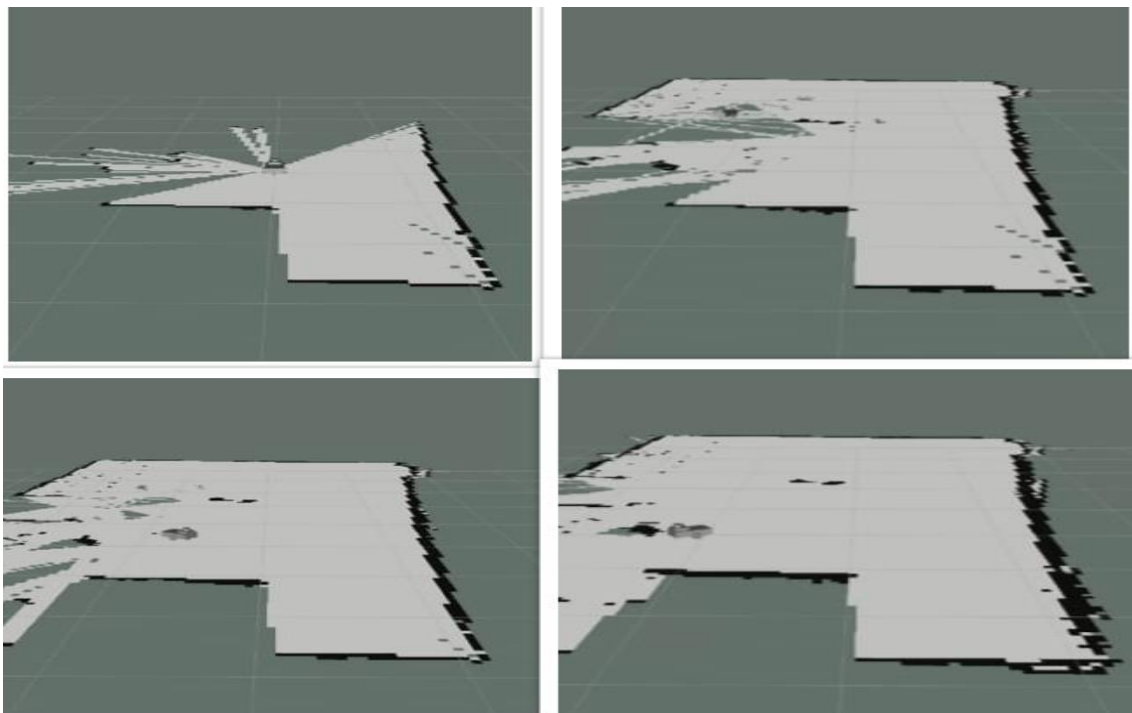


Fig 6.9: Mapping of the Environment

Figure 6.10 is the YAML (Yet Another Markup Language) PNG of the environment used for the navigation stack.

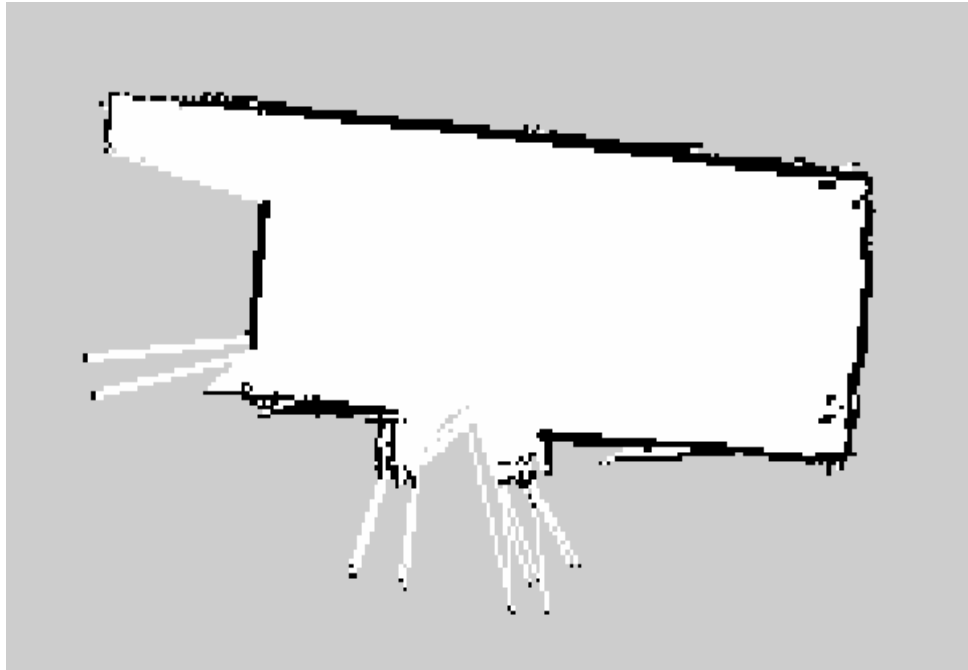


Fig 6.10: YAML Map Image

After providing the YAML to the navigation stack close the all the ROS packages. Again starting the LiDAR, RViz, URDF using Linux Terminal commands, navigation stack will be started using Linux Terminal command mentioned in the Chapter 5.

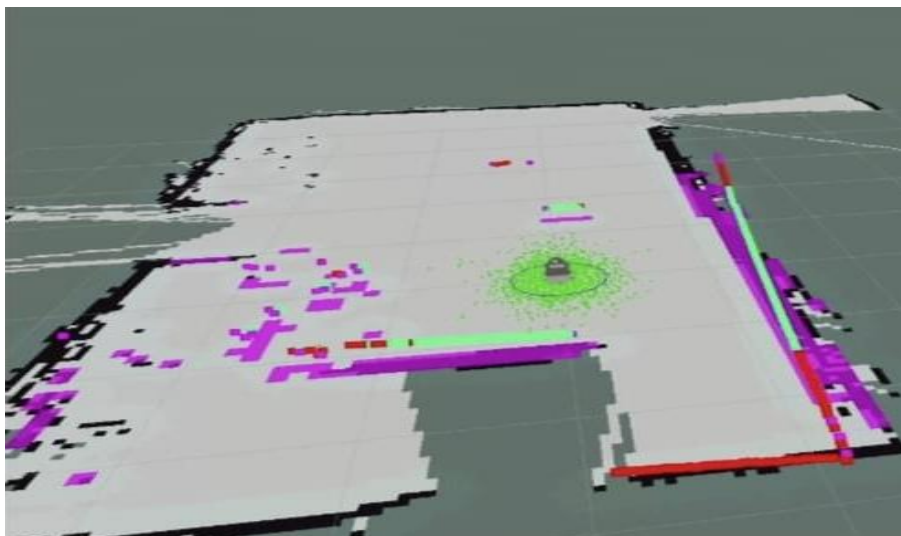


Fig 6.11: Navigation Environment

Fig 6.11 depicts the navigation stack in Rviz environment. Once we provide the destination location in the Rviz, the robot starts moving to the destination and it can be visualised using Rviz Figure 6.12.

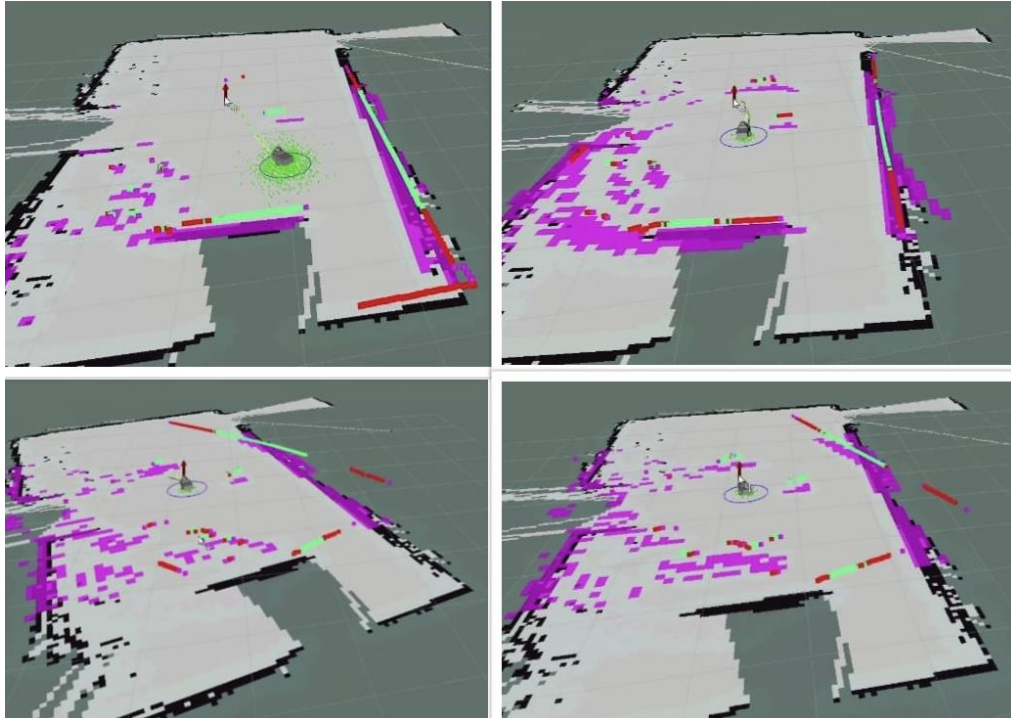


Fig 6.12: Navigation Visualisation in RViz.

Figure 6.13 depicts the robot motion towards the destination in real time environment with avoiding obstacles.

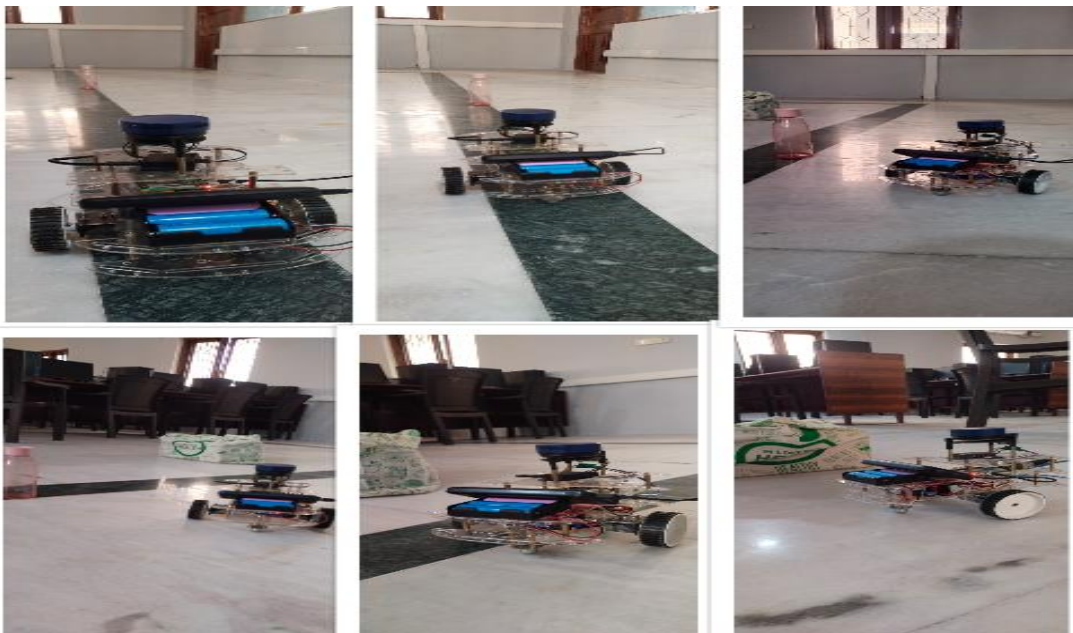


Fig 6.13: Real Time Navigation Output

CHAPTER 7

CONCLUSION

This chapter comprises of the summary of the project with the conclusion, future work, research issues and implementation issues.

7.1 CONCLUSION

In this project, we proposed a system for autonomous robot navigation using the ROS operating system and the SLAM gmapping technique with RViz and the Gazebo toolbox. The system utilized a 2D Lidar sensor to map the environment and perform obstacle avoidance and boundary detection in real-time. The results of the simulation were found to be satisfactory in terms of efficiency, demonstrating the system's ability to navigate through an indoor environment accurately and efficiently. To further improve the accuracy of the system, the Lidar data collected in ROS was integrated with MATLAB for data processing. This integration provided more information about boundary detection and obstacle avoidance, enhancing the system's performance. The real-time environment testing of the system showed that it performed more accurately in indoor environments than outdoor environments. This is likely due to the differences in the environments' characteristics, with indoor environments being more controlled and predictable than outdoor environments. Overall, the integration of ROS and the MATLAB environment provided valuable insights into the system's performance, enabling the identification of areas that required improvement before deploying the system in a real-world scenario. The proposed system has significant potential for application in various fields, including robotics and automation.

7.2 FUTURE WORK

Future work could focus on improving the system's performance in outdoor environments, as well as exploring new applications and use cases for the technology. Further research could also investigate the potential for integrating other sensors, such as cameras and ultrasonic sensors, to enhance the system's performance and accuracy. And also, developing new algorithms for the system to enhance its performance and accuracy. For instance, machine learning algorithms

could be used to improve the system's object detection capabilities and enable it to recognize new objects that were not present in the training data.

7.3 RESEARCH ISSUES

Road boundary and obstacle detection using 2D Lidar has several research issues, including accuracy and reliability of data, processing and analysis, environmental conditions, integration with other sensors, and cost and scalability. These challenges affect the performance of the system, with accuracy and reliability being key concerns. Processing and analyzing the data collected by 2D Lidar sensors in real-time can also be complex, requiring advanced algorithms and computational power. Additionally, environmental conditions can impact the performance of the system, and the integration of multiple sensors can increase the complexity and cost of the system. Addressing these research issues is essential for the development of robust and reliable road boundary and obstacle detection systems.

7.4 IMPLEMENTATION ISSUES

Implementing road boundary and obstacle detection using 2D Lidar in outdoor environments poses several mechanical functionality challenges, such as limited field of view, weather conditions, and vibration and shock resistance. Additionally, the position of the 2D Lidar on the robot can impact the accuracy and reliability of the system. While 3D Lidar provides more detailed and accurate data, it is also more expensive and may require more power to operate. The mechanical functionality of integrating 3D Lidar into a robot for outdoor use can be challenging. Careful consideration of these factors is necessary to ensure reliable and accurate performance of road boundary and obstacle detection systems.

REFERENCES

- [1]. Demir, Sinan Özgün, Tayfun Efe Ertop, Ahmet Bugra Koku, and Erhan İlhan Konukseven, "An adaptive approach for road boundary detection using 2D LIDAR sensor", 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), IEEE, 2017, 206 – 211.
- [2]. Fouad, Ayman M and Sharkawy, RM and Onsy, Ahmed, "Fixed obstacle detection for autonomous vehicle", 2019 IEEE Conference on Power Electronics and Renewable Energy (CPERE), 2019, 217 – 221.
- [3]. Han, J and Kim, D and Lee, M and Sunwoo, M, "Road boundary detection and tracking for structured and unstructured roads using a 2D lidar sensor", International Journal of Automotive Technology, 5, 2014, 611 – 623.
- [4]. Harik, El Houssein Chouaib, and Audun Korsæth, "The heading weight function: A novel LiDAR-based local planner for nonholonomic mobile robots", Sensors, 19.16, 2019, 3606.
- [5]. Huang, Shan and Huang, Hong-Zhong and Zeng, Qi and Huang, Peng, "A Robust 2D Lidar SLAM Method in Complex Environment", Photonic Sensors, 12.4, 2022, 220416.
- [6]. Iqbal, Ayesha, "Obstacle detection and track detection in autonomous cars", Autonomous Vehicle and Smart Traffic, 2020.
- [7]. Ismail, R and Omar, Z and Suaibun, S, "Obstacle-avoiding robot with IR and PIR motion sensors", IOP Conference Series: Materials Science and Engineering, 152.1, 2016, 012064.
- [8]. Jain, Sanjyoti and Malhotra, Isha, "A review on obstacle avoidance techniques for self-driving vehicle", International Journal of Advanced Science and Technology, 29.06, 2020, 5159 – 5167.
- [9]. Janda, Florian and Pangerl, Sebastian and Lang, Eva and Fuchs, Erich, "Road boundary detection for run-off road prevention based on the fusion of video and radar", 2013 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2013, 1173 – 1178.
- [10]. Juan An, Hairong Mou, Rui Lu and Youbing Li, 6th International Conference on Electronic Technology and Information Science, 1827. 012089, 2021.

- [11]. Madhavan, TR and Adharsh, M, " Obstacle detection and obstacle avoidance algorithm based on 2-D RPLiDAR", 2019 International Conference on Computer Communication and Informatics (ICCCI), 2019, 1 – 4.
- [12]. Rakotondrajao, Fabien and Jangsamsi, Kharittha, "Road boundary detection for straight lane lines using automatic inverse perspective mapping", 2019 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), IEEE, 2019, 1 – 2.
- [13]. Yalcin, O and Sayar, Ahmet and Arar, OF and Akpinar, Samet and Kosunalp, Selahattin, "Approaches of road boundary and obstacle detection using LIDAR", IFAC Proceedings Volumes , 46.25, 2013, 211 – 215.
- [14]. Yan, Kai and Ma, Baoli, "Mapless navigation based on 2D LIDAR in complex unknown environments", Sensors, 20.20, 2020, 5802.
- [15]. Zhang, Xuexi and Lai, Jiajun and Xu, Dongliang and Li, Huaijun and Fu, Minyue, "2d lidar-based slam and path planning for indoor rescue using mobile robots", Journal of Advanced Transportation, 2020,2020, 1 – 14.
- [16]. Zhu, Xuekui and Gao, Meijuan and Li, Shangnian, "A real-time road boundary detection algorithm based on driverless cars", 2015 4th National Conference on Electrical, Electronics and Computer Engineering, 2015, 843 – 848.

APPENDIX A

IEEE STANDARDS FOR LIDAR SENSOR

The IEEE (Institute of Electrical and Electronics Engineers) has published several standards related to LiDAR (Light Detection and Ranging) sensors.

- **IEEE 1547.4** - This standard provides guidelines for the interconnection and interoperability of LiDAR systems used for grid management and control. It covers topics such as communication protocols, data formats, and performance requirements. The standard aims to ensure that LiDAR systems used in power systems are reliable and effective, and that they can communicate with other devices in the power system in a standardized way. This standard is important for the integration of renewable energy resources into power systems, as LiDAR systems can be used to measure wind and solar resources and help manage their integration into the grid.
- **IEEE 1455** - This standard provides guidance on the selection, evaluation, and testing of LiDAR sensors used in remote sensing applications. It covers topics such as accuracy, precision, reliability, and stability of LiDAR sensors, and provides methods for evaluating their performance. The standard is important for ensuring that LiDAR sensors used in remote sensing applications, such as atmospheric sensing or topographic mapping, are reliable and accurate. The standard provides guidance on how to evaluate the performance of LiDAR sensors, which is important for ensuring that data collected from these sensors is accurate and can be used for scientific research or other applications.
- **IEEE 1878** - This standard provides guidelines for the use of LiDAR sensors in autonomous vehicles. It covers topics such as LiDAR system design, performance testing, and safety considerations for autonomous vehicle operation. The standard aims to ensure that LiDAR systems used in autonomous vehicles are safe, reliable, and effective. This is important for the development of autonomous vehicles, as LiDAR sensors are an important part of the sensing and perception systems used by these vehicles. The standard provides guidance on how to design LiDAR systems for autonomous vehicles, how to test their performance, and to ensure that they operate safely in various driving conditions.

- **IEEE 2147-2020:** Standard for Safety and Performance of Light Detection and Ranging (LIDAR) Systems Used for Active Safety in Automotive Applications - This standard provides guidance on the design, testing, and deployment of lidar systems used in active safety applications in automobiles. It covers topics such as system safety, performance requirements, and testing procedures. The standard provides a framework for assessing the safety and performance of lidar systems used in active safety applications, such as collision avoidance or pedestrian detection. It sets out specific requirements for the performance of the lidar system, such as its range, resolution, and accuracy, as well as requirements for testing and validation of the system.
- **IEEE 2020-2015:** Standard for Automotive System Engineering - This standard provides a framework for the engineering of automotive systems, including those that use lidar sensor technology. It covers topics such as system architecture, requirements engineering, and validation and verification. The standard provides guidance on the engineering of complex automotive systems, including those that use lidar sensor technology. It sets out best practices for system architecture, requirements engineering, and validation and verification, to ensure that automotive systems are designed and built to meet their intended functionality and performance.
- **IEEE 1516-2010:** Standard for Modeling and Simulation High Level Architecture - This standard defines a standard for modeling and simulation of distributed systems, including those that use lidar sensor technology. It provides a way for different components of a system to interact and communicate in a virtual environment, allowing for testing and validation of the system before deployment. The standard provides a framework for modeling and simulating complex distributed systems, such as those that use lidar sensor technology. It enables the development of virtual environments that can be used for testing and validation of the system, prior to its deployment in the real world.
- **IEEE 2022-2015:** Standard for Safety Analysis of Systems for Automotive Applications - This standard provides guidance on safety analysis for automotive systems, including those that use lidar sensor technology. It covers topics such as hazard identification, risk assessment, and safety requirements. The standard provides guidance on safety analysis for automotive systems, including those that use lidar sensor technology. It sets out best practices for hazard

identification, risk assessment, and safety requirements, to ensure that automotive systems are designed and built with safety in mind.

- **IEEE P2846.2:** Standard for Safety and Performance of Light Detection and Ranging (LIDAR) Systems Used for Automated Driving Systems (ADSs) - This standard is currently under development and will provide guidance on the design, testing, and deployment of lidar systems used in automated driving systems. It will cover topics such as functional safety, cybersecurity, and human-machine interaction, with a focus on lidar systems that are used for level 3, 4, and 5 autonomous operation. This standard, which is currently under development, will provide guidance on the safe and effective use of lidar sensor technology in automated driving systems. It will cover a range of topics, including functional safety, cybersecurity, and human-machine interaction, and will be tailored specifically to lidar systems that are used in level 3, 4, and 5 autonomous operation.

APPENDIX B

IEEE STANDARDS FOR AUTONOMOUS VEHICLES

The Institute of Electrical and Electronics Engineers (IEEE) has several standards related to Autonomous Vehicles, which are outlined below:

- **IEEE 2846-2017:** Recommended Practice for Safety and Performance of Autonomous Vehicles - This standard provides guidance on the design, testing, and deployment of autonomous vehicles to ensure their safety and performance. It covers topics such as system architecture, functional safety, cybersecurity, and human-machine interaction. The standard provides a framework for assessing the safety and performance of autonomous vehicles, including their hardware and software components. It sets out specific requirements for the design and testing of the system, such as requirements for the vehicle's sensors, decision-making algorithms, and communication systems. The standard also addresses issues related to cybersecurity and privacy, and provides guidance on the human-machine interface, to ensure that the vehicle's operation is transparent and understandable to its users.
- **IEEE 8300-2018:** Guide for the Interoperability of Connected Vehicles Using Internet Protocol (IP) and GeoNetworking - This standard defines the technical requirements for interoperability between connected vehicles using IP and GeoNetworking protocols. It covers topics such as communication protocols, network architecture, and security. The standard provides guidance on how connected vehicles can communicate with each other, as well as with other devices and systems, using standard protocols and interfaces. It sets out requirements for network architecture, communication protocols, and security, to ensure that connected vehicles can operate safely and effectively in a connected environment. The standard also addresses issues related to interoperability and data privacy, to ensure that vehicles can communicate and share data securely and efficiently.
- **IEEE 2668-2021:** Standard for Safety of Unmanned Aerial Vehicles (UAVs) - This standard addresses the safety requirements for unmanned aerial vehicles

(UAVs) that are used for commercial or industrial purposes, including autonomous operation. It covers topics such as airworthiness, safety management, and operational procedures. The standard provides guidance on the safe operation of UAVs, including those that are used for autonomous flight. It sets out requirements for airworthiness, safety management, and operational procedures, to ensure that UAVs are designed, built, and operated safely and reliably. The standard also addresses issues related to cybersecurity and privacy, and provides guidance on the human-machine interface, to ensure that the vehicle's operation is transparent and understandable to its users.

- **IEEE 1939-2021:** Standard for the Adoption of IEEE 1528™ Standard for Email Authentication - This standard defines the technical requirements for email authentication in autonomous vehicles, which can help prevent cyberattacks. It covers topics such as email authentication protocols, security, and privacy. The standard provides guidance on how email authentication protocols can be used to secure communications between autonomous vehicles and other devices and systems. It sets out requirements for email authentication protocols, security, and privacy, to ensure that communications are secure and reliable.
- **IEEE P2846.1:** Standard for Safety and Performance of Level 3, 4, and 5 Automated Driving Systems (ADSs) - This standard is currently under development and will provide guidance on the design, testing, and deployment of advanced automated driving systems. It will cover topics such as system architecture, functional safety, cybersecurity, and human-machine interaction, with a focus on level 3, 4, and 5 automated driving systems. This standard, which is currently under development, will provide guidance on the safe and effective use of advanced automated driving systems. It will cover a range of topics, including system architecture, functional safety, cybersecurity, and human-machine interaction, with a focus on level 3, 4, and 5 automated driving systems. The standard will provide a framework for assessing the safety and performance of these advanced automated driving systems, including their hardware and software components. It will set out specific requirements for the design and testing of the system, such as requirements for the vehicle's sensors, decision-making algorithms, and communication systems.

APPENDIX C

IEEE STANDARDS FOR MAPPING AND NAVIGATION

The Institute of Electrical and Electronics Engineers (IEEE) has several standards related to mapping and navigation, which are outlined below:

- **IEEE 1873-2015:** Standard for Robot Map Data Representation for Navigation - This standard defines a format for representing map data used for navigation by robots, including autonomous vehicles. The format includes information about the environment, such as obstacles, landmarks, and terrain features, as well as information about the robot's location and orientation.
- **IEEE 802.11mc-2018:** Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 8: Location Measurement - This standard defines a protocol for measuring the location of a wireless device, such as a smartphone or other mobile device, using Wi-Fi signals. This can be used for indoor navigation and location-based services.
- **IEEE 1851-2019:** Standard for Performance Requirements and Test Procedures for Visible Light-Based Communication Systems for Roadway Lighting - This standard defines performance requirements and test procedures for visible light-based communication systems that can be used for vehicle-to-infrastructure (V2I) communication and other applications. These systems use visible light communication (VLC) technology to transmit data over the visible light spectrum.
- **IEEE 1455-2014:** Standard for Wireless Personal Communications in Public Transit Vehicles - This standard defines a framework for providing wireless communication services on public transit vehicles, such as buses and trains. It covers topics such as system architecture, security, and interoperability, and can be used for navigation and location-based services.
- **IEEE 1512-2017:** Standard for Use of IEEE 802.11™ to Transmit Information Related to Non-Real-Time Traffic - This standard defines a protocol for using Wi-

Fi signals to transmit non-real-time traffic information, such as traffic congestion or weather conditions. This can be used for navigation and location-based services in real-time.

- **IEEE 1451-2010:** Standard for a Smart Transducer Interface for Sensors and Actuators - This standard defines a protocol for interfacing sensors and actuators with electronic systems, such as those used in navigation and mapping. It includes specifications for communication protocols, data formats, and software interfaces, and can be used to ensure interoperability between different types of sensors and systems.
- **IEEE 1733-2016:** Standard for Robot Map Data Representation for Navigation in Manufacturing Settings - This standard defines a format for representing map data used for navigation by robots in manufacturing settings. It includes information about the environment, such as equipment, obstacles, and production flow, as well as information about the robot's location and orientation.
- **IEEE 1902.1-2013:** Standard for Location Services (LS) - This standard defines a framework for providing location-based services, such as navigation, tracking, and location-based advertising. It includes specifications for location technologies, data formats, and communication protocols, and can be used to ensure interoperability between different types of location-based services.
- **IEEE 802.22.3-2018:** Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Wireless Regional Area Networks Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 3: TV White Space Map Data Transmission - This standard defines a protocol for using TV white space (unused TV channels) to transmit map data and other information in areas where other wireless frequencies may be congested or unavailable. This can be used for navigation and location-based services in rural areas or other areas with limited wireless coverage.