Developmeng of an Efficient Motion Planning Algorihm For Cleaning Robot

Authors

Muhammad Faizan Ali Khan	16-MCT-20
Abdul Anan	16-MCT-24
Hamza Habib	16-MCT-65
Salman Khan	16-MCT-04

Supervisor Dr. Ahmed Nouman

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA
SUB-CAMPUS CHAKWAL

July 2020

Development of an Efficient Motion Planning Algorithm For Cleaning Robot

Authors

Muhammad Faizan Ali Khan	16-MCT-20
Abdul Anan	16-MCT-24
Hamza Habib	16-MCT-65
Salman Khan	16-MCT-04

Project Supervisor

Dr. Ahmed Noman

Chairman Signature:_____

Thesis Supervisor Signature:_____

DEPARTMENT OF MECHATRONICS ENGINEERING

UNIVERSITY OF ENGINEERING & TECHNOLOGY TAXILA,

SUB-CAMPUS CHAKWAL

July 2020

Abstract

With the advancement of innovation and technology, mobile cleaning robots are getting more attention of researchers to make life of mankind comfortable. Households of today are becoming smarter and more automated. Home automation delivers convenience and creates more time for people. There exist many cleaning robot but they have not used motion planning algorithms to clean the environment. Some robot's move randomly to clean the surroundings while some robots use SLAM techniques to clean the area but they are too much expensive. In this project, we have developed a cleaning robot and efficient motion planning algorithm in which robot follow a zig-zag trajectory. This project is accomplished on Robot Operating System (ROS) which is very modern tool for real world simulations .This project includes the robot modeling, SLAM, autonomous mapping using frontier based algorithm, map tagging, autonomous navigation and efficient planning algorithm. This cleaning robot can perform in both modes i.e. manual as well as autonomous. Frontier based Exploration algorithm is used to make autonomous map. This work can be very useful in improving life style of mankind.

Keywords - Motion planning, SLAM; ROS; cleaning; Autonomous Robot; mapping; Localization

Undertaking

We certify that work titled **"Development of an Efficient Motion Planning Algorithm For Cleaning Robot"** is our own work. The work has not been presented elsewhere for assessment. The material that is taken from other sources is properly referred.

Muhammad Faizan Ali Khan

(16-MCT-20)

Abdul Anan

(16-MCT-24)

Hamza Habib

(16-MCT-65)

Salman Khan

(16-MCT-04)

Acknowledgements

We express our aggregate accommodation and interminable appreciation to Allah Almighty, The Most Gracious, The Most Merciful, without whose kindness and beauty, this attempt couldn't be conceivable even in its smallest. We bow before our caring enrichment's. Heavenly PROPHET MUHAMMAD (Peace Be Upon Him) who is ever a light of direction and information for humankind all in all. We consider it a respect and benefit to pay our sincere thanks and appreciation to our Project Supervisor, Dr. Ahmed Nouman for his direction, exhortation and constant consolation and motivation, particularly amid hard and edgy periods of our work. He helped us at all times this monotonous assignment. No affirmation could ever satisfactorily express our commitment to our folks who have dependably wished to see us flying high up at the skies of achievement. Without their petitions, penances and consolations the present review would have been our joyful dreams.

Likewise, we would like to extend our entire hearted because of the University of Engineering and Technology Taxila and whole staff of Mechatronics Department, for giving vital help and direction aimed this venture. By and by we are grateful to our venture consultant for working under his direction. This is an advantage which will truly remain a light tower for the duration of our life.

Table of Contents

Abstract **ii**

List of Figures **viii**

List of Tables **x**

List of Abbreviations **xi**

List of Software's **xii**

1 Introduction **1**

1.1 Problem Statement 2

1.2 Contributions 3

2 State of the Art **4**

2.1 Robot Operating System (ROS) 4

2.1.1 ROS File System 5

2.1.2 ROS Computational Graph 6

2.2 Simulation & Visualization Tools 7

2.2.1 Gazebo 7

2.2.2 Rviz 8

2.3 Autonomous SLAM 9

2.4 Autonomous Navigation & Path Planning 10

3 Related Work **11**

3.1 Robot Functions 15

3.1.1 Structure of Mobile Robot 16

4 Mathematical Modeling **19**

4.1 Differential Drive 19

4.1.1 Mathematical Modeling of The Robot 19

4.2 Robot Kinematics 20

4.2.1 Forward Kinematics 21

5 Robot Modeling in ROS **27**
5.1 Robot Model . . . 27
5.1.1 Joint Types . . . 28
5.1.2 Basic Tags . . . 29
5.1.3 Solid works 3D model . . . 30
5.2 URDF Links . . . 30
5.2.1 Setting-up the Properties . . . 31
5.2.2 Chassis Link . . . 31
5.2.3 Caster Wheels Links . . . 32
5.2.4 Cliff & Wall Sensors Links . . . 33
5.2.5 IMU Link . . . 34
5.3 URDF Joints . . . 34
5.4 Gazebo Plugins Configuration . . . 35
5.4.1 Lidar Laser Scanner Plugin . . . 35
5.4.2 Differential Drive Controller . . . 35
5.4.3 IMU Plugin . . . 36
5.4.4 Joint State Publisher . . . 36
5.5 Sensors Integration . . . 36
5.5.1 LiDAR Sensor . . . 36
5.5.2 IMU Sensor . . . 36
5.6 Motion Integration . . . 37
5.6.1 Gazebo Simulation . . . 39
5.6.2 Gazebo Environment/World . . . 41

6 Simultaneous Localization & Mapping (SLAM) **43**
6.1 Mapping . . . 43
6.2 Localization . . . 44
6.2.1 Global . . . 45
6.2.2 Local . . . 45
6.2.3 AMCL . . . 45
6.3 Autonomous Mapping Using Frontier Exploration . . . 47
6.3.1 Working of Algorithm . . . 49
6.3.2 Possible Indicators . . . 49
6.3.3 Algorithm . . . 51

7 Autonomous Navigation **53**
7.1 Introduction . . . 53

7.2 A* Algorithm . . . 54
7.2.1 Algorithm . . . 54
7.2.2 A* Example . . . 54
7.2.3 A* MATLAB Animation . . . 55
7.3 ROS Navigation Stack . . . 57
7.3.1 Navigation Stack Architecture . . . 57
7.3.2 Understanding *move_base* Node . . . 58
7.3.3 Functionalities of Navigation Stack . . . 59
7.3.4 Map Tagging . . . 59
7.4 Implementation Simulation of Autonomous Navigation . . . 60

8 Motion Planning **62**
8.1 Cleaning Motion Planning Techniques . . . 62
8.1.1 Zig-Zag Motion Algorithm . . . 63
8.2 Path Plan Method . . . 63
8.3 Working of Algorithm . . . 66
8.4 Working Flow-Chart . . . 68
8.5 Final Working Simulation . . . 69
8.6 Social Impacts & Recommendations . . . 70

9 Conclusions **72**
9.1 Future Work . . . 73

References **73**

Appendices **79**
.1 Gazebo Plugins . . . 79
.2 Engineering Drawing . . . 83
.3 Code . . . 85
.4 Gantt Chart . . . 87
.5 Team Charter . . . 88
.6 Closeout Memos . . . 89

List of Figures

2.1 Robot Model 7
2.2 Gazebo Environment 8
2.3 Rviz Display 9
2.4 Path Coverage using Autonomous Navigation 10

3.1 Roomba Robot [14] 12
3.2 Braava Robot [15] 12
3.3 Neato XV-II Cleaning Robot [16] 13
3.4 Demonstration of the environment for cleaning robot position and direction [18] 15
3.5 Succession illustration of suggested framework 16
3.6 Flow chart of robot structure 17

4.1 The robot pose in x, y, and θ in the global coordinate system [23] 20
4.2 iRobot Cleaning Robot [14] 21
4.3 A single wheel of the robot rotating along the local y-axis 22
4.4 Wheel configuration for a differential drive robot 22
4.5 Detailed diagram of the differential drive system 23
4.6 Rotation of mobile robot with $\omega\delta t$ degrees around the ICC 24

5.1 Robotics structure using concept of links and joints [26] 29
5.2 Solid Works Model 30
5.3 Light detection and ranging sensor for distance measurement [29] 37
5.4 Inertia Measuring Unit (IMU) Module [30] 37
5.5 Robot URDF Model 39
5.6 Robot URDF Model 40
5.7 Gazebo World/Environment of Home 41
5.8 Gazebo World Isometric View 42

6.1 Map of World 44
6.2 Python Script For Localization 46
6.3 Localization in world 46

6.4 Accurate Probabilistic Model of Map 47
6.5 Frontiers Representation [35] 48
6.6 Mapping Through Frontier Exploration 52

7.1 A* Example [44] 55
7.2 Demo Animation of A* Before Path Estimation [45] 56
7.3 Demo Animation of A* After Path Estimation [45] 56
7.4 Architecture of Navigation Stack [47] 57
7.5 Map Tagging 60
7.6 Initial world Map in Rviz 60
7.7 Path Planning From Start to Goal State 61
7.8 width=1 61
7.9 Goal Achievement in Autonomous Navigation 61

8.1 Zig-Zag Motion Path 64
8.2 Zig-Zag Path Planning 67
8.3 Path Coverage Steps 67
8.4 Cleaning Robot Autonomous Navigation Flowchart 68
8.5 Publishing the Polygon 69
8.6 Planning and Working of Robot 69
8.7 Importance of features of cleaning robots [53] 71

List of Tables

2.1 Terms Used in ROS File System 6
2.2 Features in ROS Computation Graph 6

4.1 Description of DOF with respect to altitude 20

5.1 Join Types Description 29
5.2 Basic Tags Description 30

1 Implementation Plan 87

List of Abbreviations

ROS ..... Robot Operating System

ICC ..... Instantaneous Center of Curvature

LIDAR ..... Light Detection and Ranging

DOF ..... Degrees of Freedom

URDF ..... Unified Robot Description Format

SLAM ..... Simultaneous Localization and Mapping

AMCL ..... Adaptive Monte Carlo Localization

FBA ..... Frontiers Based Algorithm

BFS ..... Breadth First Search

CAD ..... Computer Aided Design

List of Software's

Sr.	Tool	Usage
1	Ubuntu 18.04	ROS is Compatible with Linux
2	ROS (Robot operating system)	Tools, Libraries, Simulations
3	Gazebo	Simulator
4	Rviz	visualization of robot
5	Python	algorithms scripts
6	Blender	Gazebo World
7	SolidWorks(2018)	Robot Modeling
8	Sweet Home	Gazebo Home world
9	MS Visio	Flow Charts
10	LaTeX	Presentations, documentations

Chapter 1

Introduction

"Ask not what your country can do for you; ask what you can do for your country."

John F Kennedy

A very notable household chore is floor cleaning which is often considered as unpleasant, difficult, and boring. In household robotic cleaners have made their way in mainly floor cleaning applications. Robots working in natural environment so that it provides a useful service like cleaning is in contrast to industrial robots designed to perform a specific task with repetition furthermore designing a home or office friendly robot is different considering the dynamics of their environment [1].

Cleaning requires constant attention and repetition due to its nature of difficulty thus, it is often done with the help of machine but in the strict supervision of human. The necessity of such a equipment which can assist humans arises leading to the creation of autonomous robot vacuum cleaners [2]. Vacuum cleaner is defined as electromechanical device used for cleaning floors, rugs, furniture, and carpets by sucking dust particles inside a suction chamber [3].

Robotic vacuum cleaners are mostly expert in floor mopping, dry vacuum cleaning etc. Neato and iRobot are two well known Robotic Cleaner Robots companies but very expensive [4]. These already existing models pose some serious problems in terms of their operating algorithms. For example, Roomba can clean very efficiently but it neglects the floors beneath any furniture and corners of walls [5]. To make sense of operating mech-

anism of robotic vacuum cleaner we need to examine steps for floor cleaning process. There are two steps to clean the environment. The dry cleaning is wherein the dust is cleared using suction principle [6]. Most of the robotic vacuum cleaners only focus on the dry-cleaning process but covering the whole area of house or office is difficult and thus poses the challenge for optimal solution. Currently robotic floor cleaners use variety of processes to maximize the cleaning efficiency. For the sake of paper, we discuss only two i.e. Laser Mapping technique and collision avoidance technique.

The laser mapping technique is far more effective as it eventually creates a digital map of whole area. The robots with the features of mapping and path planning are relative faster, time and energy efficient but on the downside are far more expensive. LiDAR sensor or camera helps it to scan constantly for any obstacles [1] The other technique which is comparatively cheaper only uses Collision avoidance. It bumps with the walls and turn back at certain orientation and make random motion to clean the environment but it is takes too much time to clear the whole surface as there is no mapping and navigation used in it thus their remains clue if the robot cleared the whole room or not [7, 8].

We approach this problem by getting best of both worlds. In this paper we try to create and examine more efficient algorithm which is both time saving as well as inexpensive to implement. For this purpose, we are using Robot Operating System and Simultaneous Localization and Mapping SLAM for autonomous mapping [9].

The Robot Operating System provides easy to use simulation softwares which helps to evaluate our algorithm. By using this concept this cleaning mobile robot will be able build a map of the unknown environment using a special technique and then robot will use this generated map to navigate the environment at the same time. This thesis concludes with the demonstration of fully functional algorithm for vacuum cleaning robot.

1.1 Problem Statement

"To develop an efficient algorithm for automated navigation and cleaning of environment using a mobile robot"

1.2 Contributions

The Project contribute in following catagories:

- URDF Modeling
- Gazebo Simulations
- Automated Mapping
- Autonomous Navigation
- Map Tagging
- Making an Efficient Algorithm - Path Planning

Chapter 2

State of the Art

"The world requires practical dreamers who can, and will, put their dreams into action."

Napoleon Hill

2.1 Robot Operating System (ROS)

ROS is well known by its simulations in the field of robotics and it is open-source. Robot operating system comes with interfaces of message passing, tools for simulation i.e. gazebo, rviz and have a lot of packages which help in simulating the project. Due to its several integration tools it will not make much complexity. Different libraries and packages are provided by many people to help in sharing knowledge. ROS is used to pass messages. To develop new applications for researchers it provides the good facilities. In this thesis work, ROS is the main base of work because of publishing of messages on the topics and then subscribing those messages from topics. This will be carried out with different parameters. ROS also provides Interplatform operability, Modularity, Concurrent resource handling.

For the creation of virtual environment in which simulated robot model is made to implement different algorithms ROS is mainly used. We can visualize all the things in virtual environment rather than implementing the overall system and hardware. So, we can implement every thing in simulation and then transform it to hardware. Below given are some important features of provided by ROS

- **Message passing interface:** As the name suggests it provides the capability of passing messages. The program exchange the data with the help of linked systems of communication.
- **Package management:** The ROS packages are designed with the help of ROS nodes. These packages consists of different files like source codes, config and build files etc. The build system helps to execute these packages. This developed and systematic approach of management is provides by ROS.
- **Low-level device control:** There sometime needs to control low level electronics to send data with the help of serial ports and vice versa. This can only be attainable with the help of ROS.
- **Distributed computing:** When we are interfacing different sensors then processing the data is very high from different robot sensors. In ROS, this high processing is distributed to clusters which have many computing nodes. This helps in increasing the speed of processing data.
- **Code reuse:** The importance in term of code reuse is very demanding so ROS growth factor is increasing day by day due to this sharing of executable. The group of different packages are used to run one simulation this whole is known as meta package and both of these packages are shared or ditributed.
- **Scaling:** For performing complex and difficult computations in robot ROS can be scaled.

ROS gives the combination of different messages passing interfaces, simulation and visualizations of tools, and capabilities. There are many capabilities of navigation, control, localization, manipulation and mapping etc. are available. This results in development of many robotics applications.

2.1.1 ROS File System

The organized way of files which includes different types of packages, messages etc. are known by ROS file system. The below given Table 2.1 gives the description about main terms using in file system

Term	Description
Packages	It is individual unit which contains codes, libraries, config files, executable files
Messages	The description of messgage is stored in msg in package. To send the messages, these data structures are used. The extension is .msg
Services	The srv folder in package describes the services having .srv extention. it is defined by request & response data structure.

Table 2.1: Terms Used in ROS File System

2.1.2 ROS Computational Graph

This computational graph in ros are defined by the peer-to-peer network that is used in processing the data. The features in computation graph are given in below Table 2.2

Feature	Description
Nodes	This is the executable part of package which process the data using functionalities
ROS Master	A program which is known as ros master is used to connect the ROS nodes to each other.
ROS Topics	ROS topic is used by ROS nodes for the communication among each other.
Services	Service request will be sent to other node that is responsible to provide the service with the help message definition. This result is then sent as a reply. The node will wait for this result until it received from particular node
Bags	In ROS bags data is stored so that it can be played back for ROS topics.

Table 2.2: Features in ROS Computation Graph

2.2 Simulation & Visualization Tools

2.2.1 Gazebo

Gazebo is a tool of ROS which is used to simulate a robot in the virtual environment. After the creation of robot or world it is transformed to Gazebo as a robot model and world respectively. The complex environments can be created with the help of this tool. All the sensor can be mounted on the robot to visualize the environment in 3D space. The robot model is made through URDF which we discuss in next chapter and assign different links to them accordingly. From links and joints different degree of movements can be created. A differential drive robot is used in this thesis and the mathematical modeling for that is given in next chapter. The robot consists of two wheels, two casters and lidar for scanning of environment as shown in 2.1. The sample map is shown in 2.2. In this environment, the house environment is created, having kitchen, bedroom, launch and corridor. This map includes different objects which are considered as obstacles.

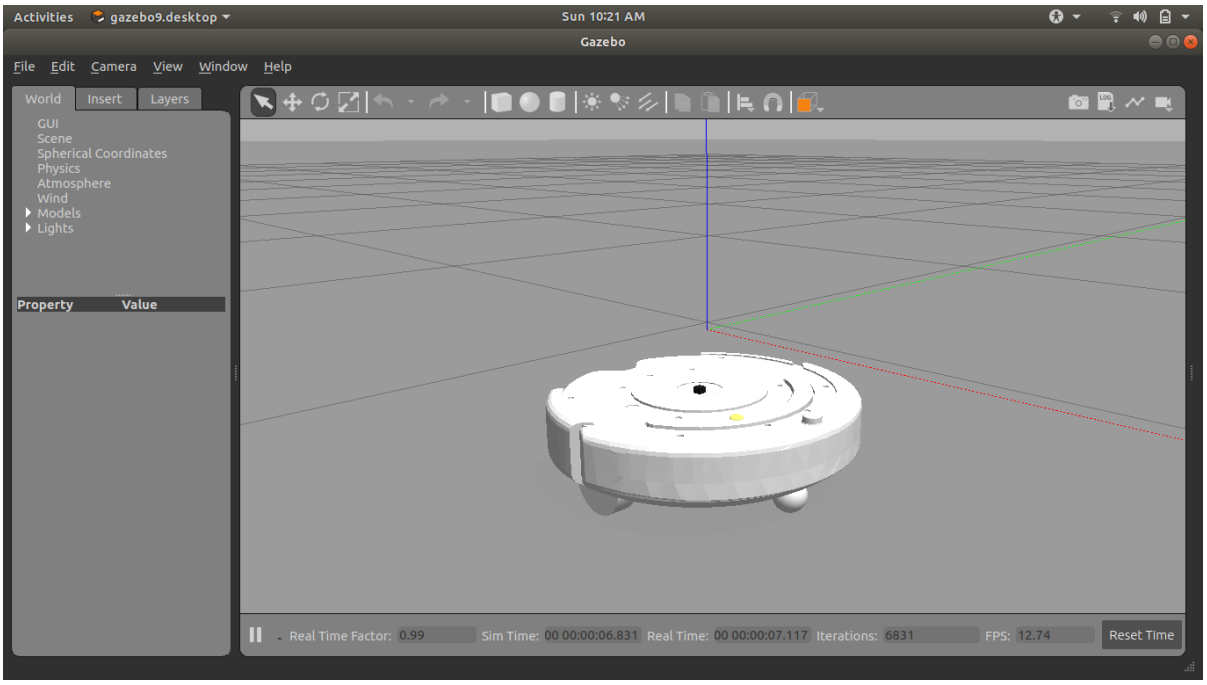


Figure 2.1: Robot Model

Gazebo is famous about its accuracy and efficiency in simulation of complex robots in indoor & outdoor environments. High quality graphics and renderings can be created with the help of gazebo.

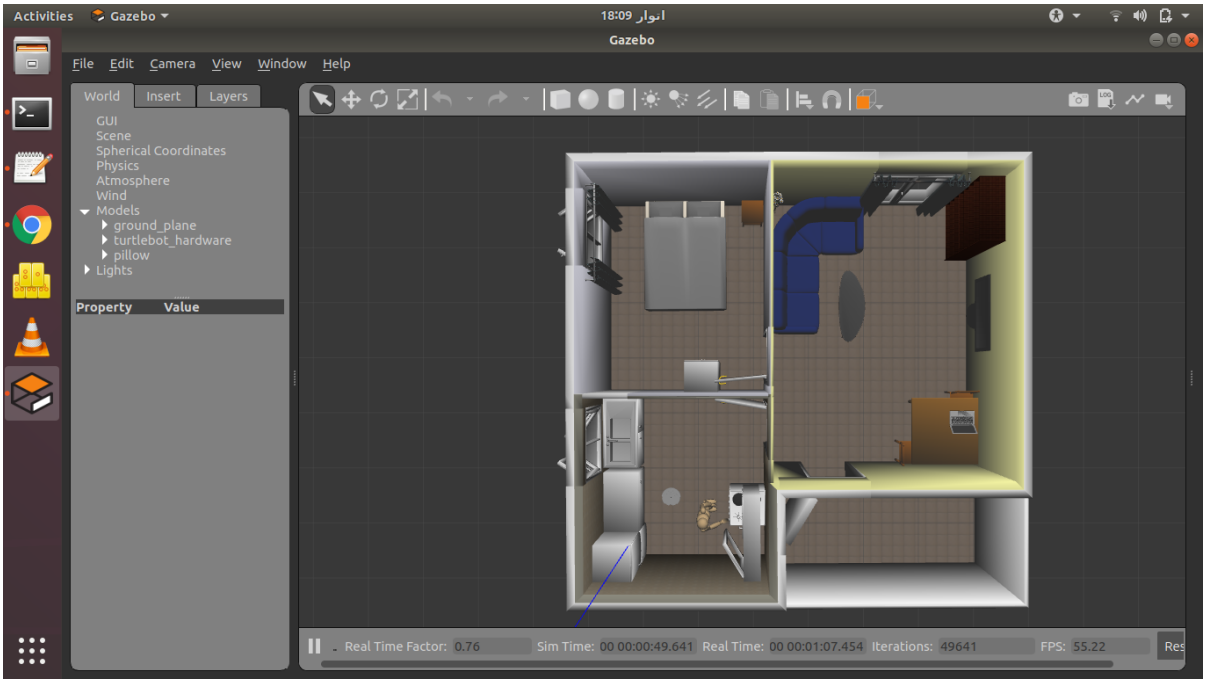


Figure 2.2: Gazebo Environment

The Gazebo Features are

- Dynamic Simulation
- Advance 3D Graphics
- Sensor Support
- Plugins
- Robot Models
- Command Line Tools

2.2.2 Rviz

Rviz is used to visualize the urdf, sensors and their data in 3D environments. Means to say if lidar is fitted on the robot in gazebo, then we can visualize its scan data with the help of Rviz. With the help of scan data map can be build and visualized and then used for autonomous navigation. This can represent IMU, laser scan etc data graphically as well. The coordinates are called frames in Rviz.

Many displays can be selected in Rviz to visualize the data from different sensors. ADD

button when clicked we can display any data from sensors. The reference and ground position will be given with the help of grid. From sensor/Laser, scan data can be displayed. The position given by program is displayed by Point Cloud. Axis can be displayed with the help of reference point. The initial display of Rviz is given below in Figure 2.3

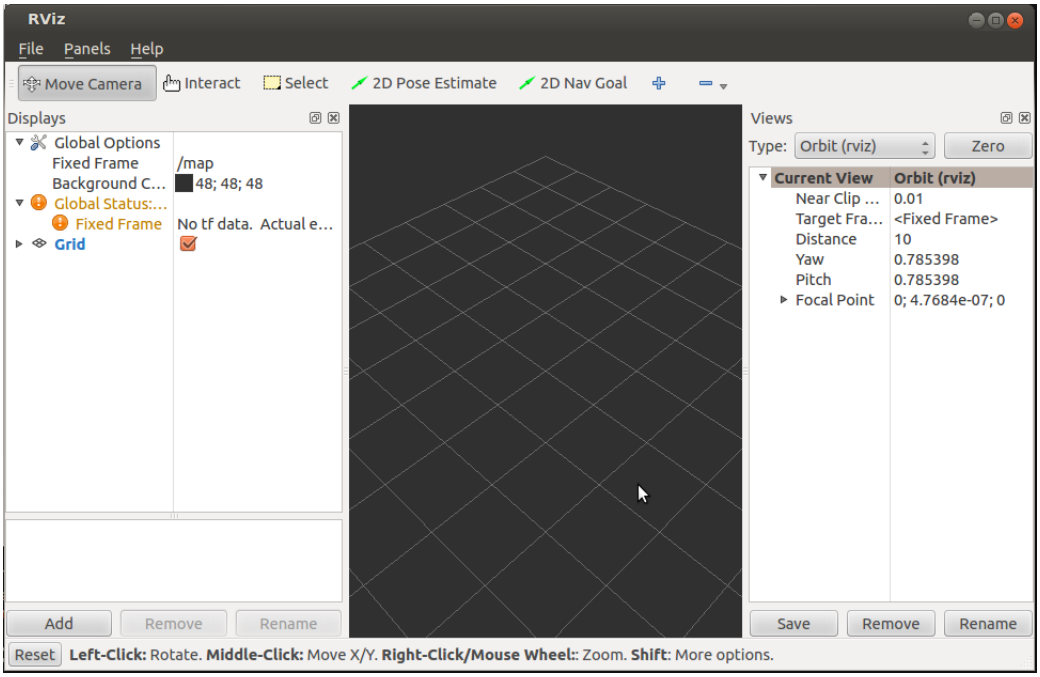


Figure 2.3: Rviz Display

2.3 Autonomous SLAM

The primary aim of the automated robot model is finished the SLAM and then indoor SLAM needs the automated cleaning vehicle to begin operating from an unspecified location and construct map by evaluating the location throughout the development. Simulation constructing analysis environment and convey movement command data through the computer console to handle the automated cleaner to shift in environment. The autonomous mapping is carried out through the exploration technique known as frontier based technique in which frontiers are search one by one to create map of the environment.

2.4 Autonomous Navigation & Path Planning

After generating the map of environment, we check the practicality and adequacy of automated cleaning robot navigation, determine the goal situation in ROS visualization tool like Rviz. ROS Navigation proceed the present location. then goal location are joined with map to generate a global path way, as well as indicated by the current movement velocity and bearing in the development of robot, the ideal velocity of every movement cycle is managed, as the local path plan.

When a global plan is generated the local planner makes this path into velocity orders for robot's actuators. It is done by making a worthy justification all over the robot, examining and simulating directions inside this capacity, getting every simulated direction dependent on its normal result, conveying the most noteworthy getting direction as a velocity command to the automated cleaning robot and rehashing until the achievement of goal.

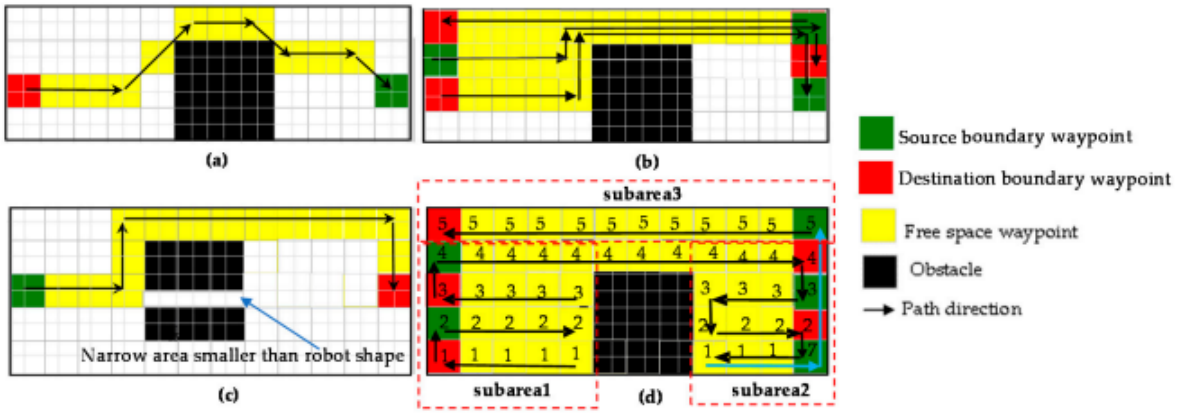


Figure 2.4: Path Coverage using Autonomous Navigation

Chapter 3

Related Work

"Dont learn everything... if you want to reach your goal, just learn what is needed."

Erik Zamora

Automated cleaning robot have grasped significant attention in mechanical robotic research due to their sufficiency in supporting people for cleaning application such as at hospitals, rooms, restaurants, working area, health center, stockrooms, universities and so on. Automated cleaning robots are known by their suitable competence like floor polishing, dried vacuum cleaning and so forth [10]. Robot vacuum cleaner is a self-governing automated machine which incorporates automated mode and polish the surface independently without aid of human. This automated cleaner comprises of sweepers, wiping, Ultraviolet sanitization, surveillance cameras due to cleaning applications [11]. Automated robotic cleaner should be equipped with sensors and use way arranging calculation must have the option to finish the assignment with great outcomes. The path arranging calculations ought to be complex so as to make educated assumptions about the working condition and have th e option to respond to a changing indoor condition. The branch of knowledge of path arranging is particular because of the quantity of factors that influence how well a mechanical vacuum cleaner can clean [12]. A part of the available cleaning robot classifications is considered below: **iRobot** Corporation is an American innovation organization that structures and fabricates buyer robots. This organization items incorporate self-governing vacuum cleaner (Roomba), floor moppers (Braava), floor washing robots (Scooba) and different self-sufficient cleaning items.

1. ***Roomba***: Roomba robot shown in Figure 4.2 was propelled in 2002 by American organization. It includes IR, RF and auto charging technique is utilized for dry vacuum reason [13]. Roomba robot is capable to navigate surface area of place and perform cleaning action. Roomba consist of energy conserving features and turn of indicating light while charging itself in order to conserve energy.

Figure 3.1: Roomba Robot [14]

2. ***Braava***: Braava iRobot's shown in Figure 3.2 was launched in 2006 and its main application was to clean hard surfaces. It use replaceable cloth to clean the surface of floor. Infrared including effective wall supplement technology is used for cleaning purpose wall [10].

Figure 3.2: Braava Robot [15]

Neato Robotics is an autonomy organization their most popular items are Neato-Robots XV arrangement. With the innovation of cleaning robot numerous nations began fabricating cleaning robots. China additionally fabricated cleaning robots with progressively advance component and solid innovation.

1. ***Neato XV-11***: It is vacuum cleaning robot shown in Figure 3.3 including Laser span discoverer innovation, SLAM (Simultaneous limitation and planning) and auto-charging innovation but no path planning and too much expensive [10].

Figure 3.3: Neato XV-II Cleaning Robot [16]

The overall ROS people group unites all significant information and an assortment of robot-important open-source programming on a 'normalized' stage, or middleware, running on Linux.The primary features of ROS are summarized below:

1. **Point to point design:** ROS is certainly not a single procedure framework, yet a performing multiple task, multi-process framework. Accepting that robots are consist of on these kinds of system, then the robot hardware requirement will be very high and resources will be wasted. This design can suburbanize the work load and the undertaking of suburbanized is especially noticeable in multi machine correspondence.

2. **Multi-language support:** ROS includes a working framework which is unbiased to the different languages. It can support languages like C++, Python, LISP, and Octave, and other to interface with ROS environment. ROS includes feature of blending programming, which basically converts into handling design for information that empowers numerous languages to show qualities and supplement one another.

3. **Streamlined and integrated:** Streamlined and coordinated Modular is consequence ordinary for ROS. Code of every module is arranged independently. Incorporating just needed composing relating CMake documents.

4. **Rich toolkit:**To deal with the perplexing ROS programming structure, there are different three-dimensional representation troubleshooting programming like Rviz to encourage advancement of automated cleaning robot. Furthermore, the framework node operation and procedure boundaries are straightforwardly seen in Rqt. Using ROS' circulated handling structure, plans an elite portable robot stage for indoor SLAM and acknowledges independent route work on the stage [13].

The task, in day by day life, is the assignments which is taken, obligation and duties are required to focus on. In area of PC, it alludes to process and progression exchange, finished by PC, just as the expert words of fundamental load unit. we characterize assignment by launch. As a matter of first importance, a couple of expert terms are presented as follows.

ROS node: A node is practicable document, which can communicate with ROS condition. It communicates through ROS Master with different nodes. **Launch file:** ROS utilizes the device - "roslaunch" to start the document. Launch file is written as XML, incorporates essential nodes. Launch record proceed, which start the instrument - "roscore" and one-time start the group numerous node one-time which is characterized in the launch document. **Package:** Package, is the most fundamental unit association with ROS code. Packages incorporate library documents, practicable records, contents, and different documents. At the stage when a node is set up, the node is looked by the package where it exists.

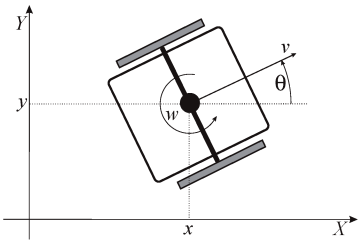
In ROS environment, practicable record of a node is characterized in package, where different practicable documents lie. At that point, cluster various node pursues beginning the dispatch document where the hub list is characterized throughout the device - "roslaunch". Thus, a launch record characterizes an assignment for the automated machine to finish. Beginning a dispatch record implies beginning an assignment. Each bundle, obviously, there can be numerous launch documents with the goal that various undertakings, comprised of various nodes, can execute. Because of the group beginning of nodes,

tasks of the robot are improved. In this way, it is sensible that composing the node list into launch documents and characterizing a launch record as an errand [17].

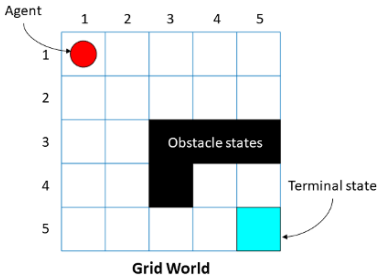
3.1 Robot Functions

The primary capacity of a self-driving robot is route. This can be characterized as the blend of the three crucial capabilities:

- Self-localizations
- Path planning
- Map-building and map interpretation



(a) Robot Pose



(b) Grid Based Representation of Environment

Figure 3.4: Demonstration of the environment for cleaning robot position and direction [18]

Localization is technique by which a robot chooses its particular region in unspecified area. Two main situations are required if robot needs to get itself: Firstly, automated cleaning robot is outfitted with various sensors that assist in detecting unspecified state of exterior world; secondly, Automated cleaning robot should be capable to generate map of unspecified environment then managed the sensor type according to their sensors [13].

Exact localization of robot is needed to generate map of unspecified environment just as satisfactory robot control rotary encoder which are used for robot odometry. Be that as it may, an estimation blunder is gathered after some time, which brings about a signicant float. To upgrade localization by distinguishing noticeable highlights SLAM calculation is used. The SLAM calculation depends on a Rao-Blackwellized Particle Filter method, which empowers grid map investigation for laser search information. To make sure that

total analysis has been completed, we show a ne-grid based portrayal of the surrounding S see Figure 6.6b. Approximating surrounding by cells with a same size and geometry, each free cell will proceed in any event once for complete analysis [19].

3.1.1 Structure of Mobile Robot

Working with self-ruling portable robots includes overseeing and synchronizing four principle errands: seeing the environment, localizing the robot inside this environment, arranging the following movement, and executing the arranged assignment. Figure 3.5 represents the fundamental succession of the proposed framework, which will be clarified in the accompanying areas in more detail [6].

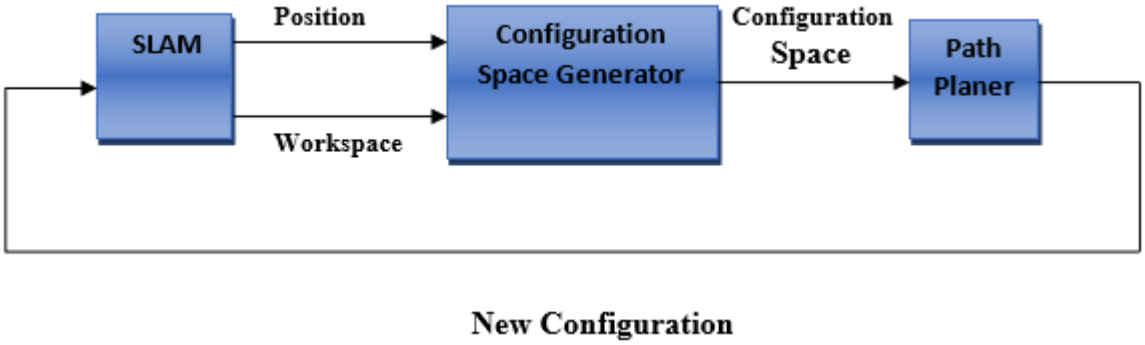


Figure 3.5: Succession illustration of suggested framework

SLAM algorithm calculates the present position and direction of the automatic cleaner and produces a map of the seeing environment (workspace). A conguration space generator permits proficient path planning algorithm that computes the following leading conguration for the automated cleaning robot to navigate to.

Rao-Blackwellized molecule channel is utilized to indicate map of automated cleaner in 2DSLAM, and the pose of the automated cleaner is appeared by various heavy molecule sets [19]. The pose assessment is executed by the cleaners movement design to be specific the positioning issue, and afterward finish the development of surrounding map. The algorithm stream of the automated vehicle investigative stage is shown by Flowchart 3.6

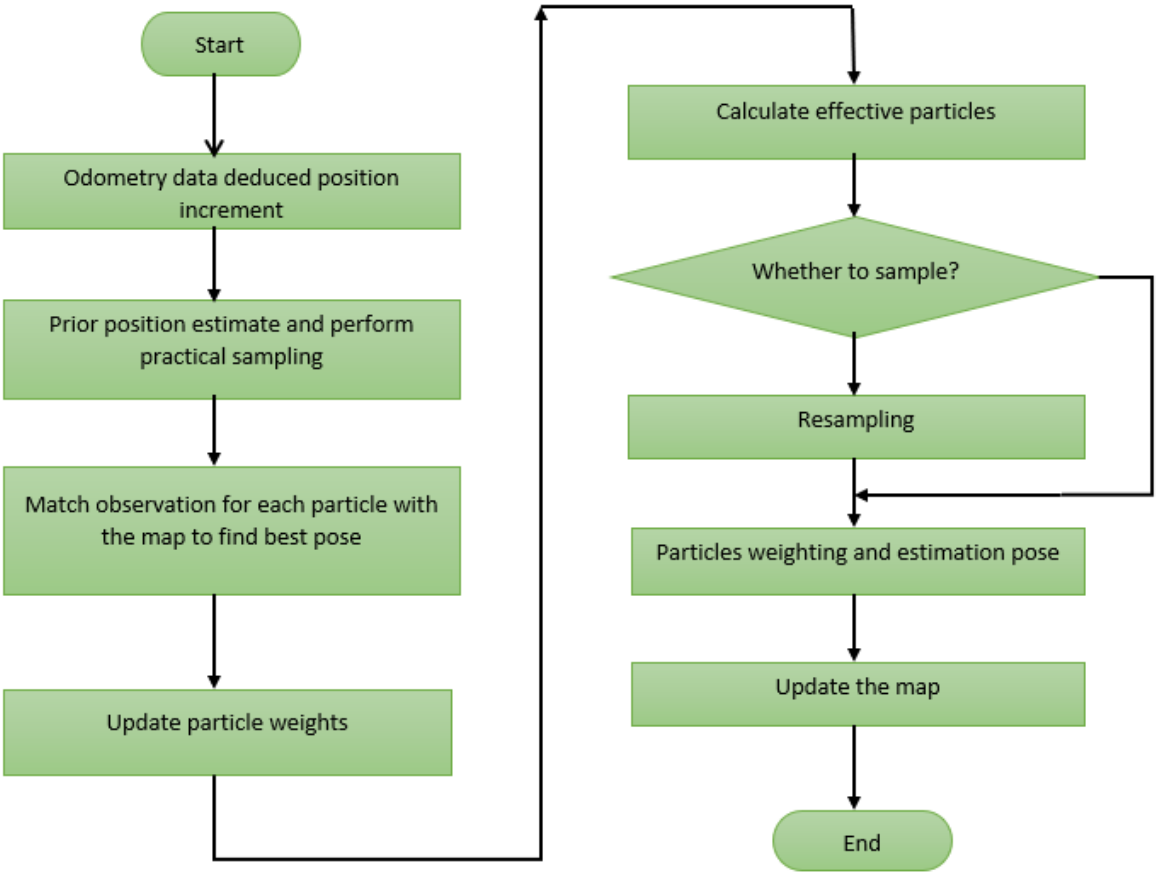


Figure 3.6: Flow chart of robot structure

The robot develops a 2D geometric portrayal of its surrounding utilizing the laser scanner. Through the wheel encoders laser scanner blend of geometric information and odometry data to decide its present location. It creates a point cloud where every point relates to a location where laser scanner trusts it could be founded on accessible information. As the automated cleaning robot moves, it precludes feasible locations and the quantity of points in the cloud diminishes. Along these lines, its number of conviction states quickly merges to its actual location. In this way, the robot accomplishes localization through probabilistic induction.

When all is said in done, the group didn't get quantifiable proportions of the cleaners exhibition, rather depending on abstract decisions of appropriateness for automatic navigation at different phases of execution. Basically, moving the robot and building visualization review of presentation, or providing the navigation commands and seeing its conduct, were adequate techniques for execution assessment all through the majority

of the task. The accompanying table sums up the group's abstract sense of the cleaner's performance at significant achievements by undertaking dependent on constant trial and perception [20].

After generating the map of environment, we check the practicality and adequacy of automated cleaning robot navigation, determine the goal situation in ROS visualization tool like Rviz. [13].

When a global plan is generated the local planner makes this path into velocity orders for robot's actuators. It is done by making a worthy justification all over the robot, examining and simulating directions inside this capacity, getting every simulated direction [20].

Chapter 4

Mathematical Modeling

"More gold has been mined from the brains of men than has ever been taken from the earth."

Napoleon Hill

4.1 Differential Drive

In previous chapter, we have discussed the basics about ROS its different tools like gazebo simulator, Rviz for visualizations and also different file systems. As we are working on the differential drive cleaning robot. having two wheels on the opposite sides of robot chassis. The direction and motion of robot is controlled via motion of these wheels.

In this chapter, we are going to give an idea about analyzing the differential drive of the robot mathematically and also solve the kinematics equation. This kinematic equation leads to help a person to predict the robot position using its sensors data.

4.1.1 Mathematical Modeling of The Robot

The steering system is the most important part of the mobile robot. This system takes part in navigation of robot from one place to other. The system which is very cost effective and simple is differential drive system. This system includes two main wheels which are here labeled by left and right wheels which are mainly mounted on the same axis. These both wheels are connected to separate motors and this drive system i.e. differential

drive system is known as nonholonomic system which is defined as the system which has constraints to change to robot pose [21].

4.2 Robot Kinematics

The kinematics of robot deals with the study of motion in mathematics without considering the force which affect the motion of robot. The geometric relationships are basically taken into concern that govern the whole system [22]. The system in which all the motion are taken into concern are known as the study of robot dynamics.

There are total 6 DOFs (Degrees of freedoms) involved in a mobile robot. They are expressed by $x, y, z, roll, pitch, yaw$. Three of them i.e. x, y, z are for position and other i.e. roll, pitch, yaw are known for altitude. The rotations are expressed in Table 4.1

DOF	Description
roll	sidewise rotation
pitch	forward and backward rotation
yaw	it is for direction in which the robot moves in xy-plane

Table 4.1: Description of DOF with respect to altitude

As the differential drive mobile robot moves horizontally in the plane from x to y and has pose which has mainly three components i.e. x, y and θ. Where theta heads towards the robot's forward direction. Below Figure 4.1 shows the information which describes the differential drive system pose

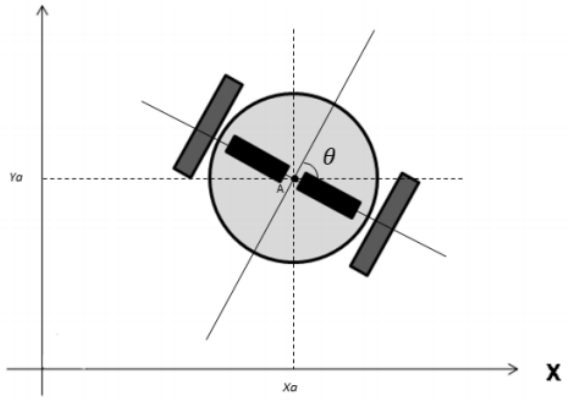


Figure 4.1: The robot pose in x, y, and θ in the global coordinate system [23]

There are two motors in mobile robot and its motion is controlled by setting up the

velocities of these motors. Here it is named by V_{left} and V_{right}.The following Figure 4.2 shows a of popular differential drive cleaning robot available on the market:

Figure 4.2: iRobot Cleaning Robot [14]

The above robot is popular robot from romba series of robots given by well known cleaning robots company iRobot.

4.2.1 Forward Kinematics

The following problem is solved by the forward kinematics equation of mobile robot with differential drive system:

Determine the pose (x', y', θ'), If mobile robot position is (x, y, θ) at time (t), and at time $t + \delta t$ the control parameters of V-left and V-right

The Figure 4.3 shows the simple illustration of single wheel to describe its local axis. This forward kinematics technique is formulated to get a solution which is calculated by mobile robot for following a particular trajectory.

The motion which is around the y axis is called as roll and other all the things are known as slip. Suppose there is no slip in our case. As we know that when a wheel cover its full rotation, the distance covered by it is $2\pi r$. where r represents the radius of the wheel. In our simulation the motion will be in 2D i.e. two dimensional which shows that surface is flat and even [24].

The rotation of robot is must along the common axis of left and right wheel to perform a turning action. ICC i.e. instantaneous center of curvature is the point about which the

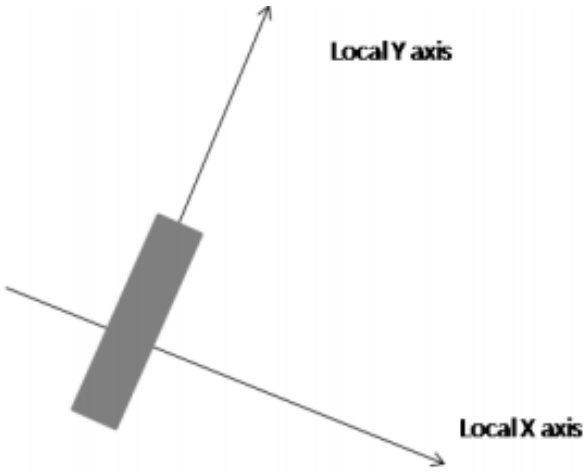


Figure 4.3: A single wheel of the robot rotating along the local y-axis

robot rotates. The location of ICC is outside the mobile robot. The wheel configuration is shown by following Figure 4.4 with respect to ICC.

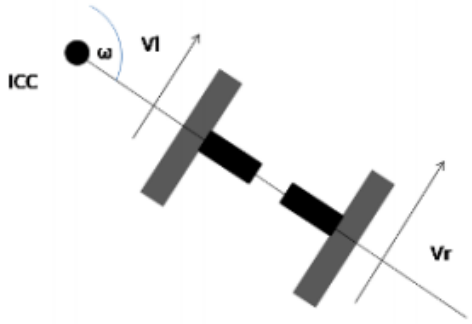


Figure 4.4: Wheel configuration for a differential drive robot

The angular velocity which is represented by ω gives the central concept to derive the kinematic equation. The rotation of each wheel is around the ICC with the radius r along the circumference.

The speed (v) of one of wheel of the wheeled robot is $v = \frac{2\pi r}{T}$, where time taken to cover the one whole rotation is represented by T. The angular velocity (ω) is represented mathematically as $\frac{2\pi}{T}$, The unit of ω is radians or degrees per second. If we combine the both v and ω equations it results in the following equation of linear velocity

$$v = r\omega \tag{4.1}$$

In Figure 4.5, the whole model of differential drive system is given

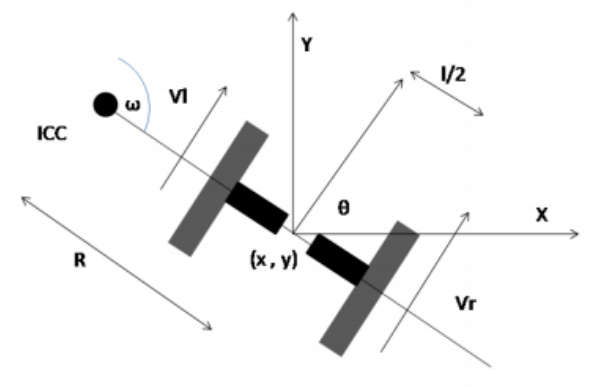


Figure 4.5: Detailed diagram of the differential drive system

The result will be same if we apply the equation on both of the wheels that is ω

$$\omega(R + \frac{l}{2}) = V_r \tag{4.2}$$

$$\omega(R - \frac{l}{2}) = V_l \tag{4.3}$$

Where R and l represents the distance b/w ICC and midpoint of wheel and length of wheel respectively. When we solve the ω and R, the following result comes

$$R = \frac{1}{2}\frac{(V_l + V_r)}{(V_r - V_l)} \tag{4.4}$$

$$\omega = \frac{(V_r - V_l)}{l} \tag{4.5}$$

To find the distance between instantaneous center of curvature to mid of robot and angular velocity we have to use this equation.

The above given equation is used to solve the forward kinematics problem. If the robot is moving with velocity of ω for time δt seconds. It will change the robot orientation and the changed heading is represented by following equation.

$$\theta' = \omega\delta t + \theta \tag{4.6}$$

With the help of basic trigonometry, we can represent center of the ICC rotation.

$$ICC = [ICC_x, ICC_y] = [x - Rsin\theta, y + Rcos\theta] \tag{4.7}$$

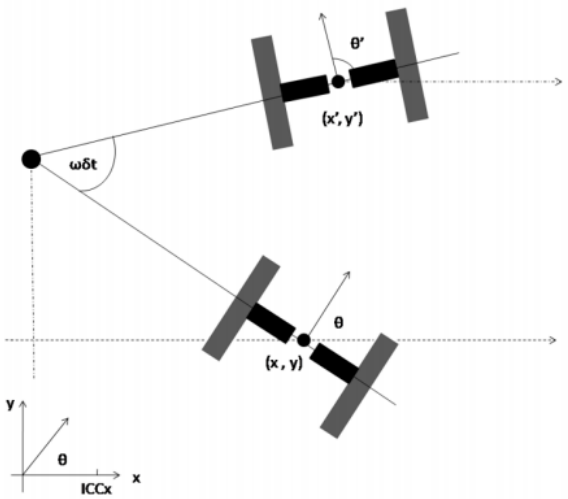


Figure 4.6: Rotation of mobile robot with $\omega\delta t$ degrees around the ICC

The starting position is given by (x, y) and the new position of the robot (x', y') can be computed with the help of rotation matrix. The rotation of robot around instantaneous center of curvature with the angular velocity of ω for time δt seconds results the position given below at the time $t + \delta t$:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos(\omega\delta t) & -sin(\omega\delta t) \\ sin(\omega\delta t) & cos(\omega\delta t) \end{pmatrix} \begin{pmatrix} x - ICC_x \\ y - ICC_y \end{pmatrix} + \begin{pmatrix} ICC_x \\ ICC_y \end{pmatrix} \tag{4.8}$$

The new positon of robot is represented by (x', y', and θ') and mathematically can be computed with equations (4.6) and (4.8), given $\omega, \delta t$, and R.

The angular velocity ω which can be calculated mathematically from equation (4.5); V_r and V_l are very difficult to measure with high accuracy. Instead with the help of wheel encoders the rotation of each wheel will be measured. The data coming from wheel encoders is known as robot odometry data [23]. Encoders are mounted on wheel axes with the shaft of motor and these deliver binary signals in result.

A counter counts these signal so that $v\delta t$ is the total distance moved from the time t to $t + \delta t$.

$$n \times step = v\delta t$$

The linear velocity (v) from encoder data can be calculated from this

$$v = \frac{n \times step}{\delta t} \tag{4.9}$$

If we insert equation (4.9) in equations (4.3) and (4.4) to calculate R

$$R = l/2\frac{(V_l + V_r)}{V_r - V_l} = l/2\frac{(n_l + n_r)}{(n_r - n_l)} \tag{4.10}$$

$$\omega\delta t = \frac{(V_r - V_l)\delta t}{l} = \frac{(n_r - n_l) \times step}{l} \tag{4.11}$$

Where n_l & n_r represents the encoder counts coming from left and right wheels motors. V_l & V_r represents the velocities of left and right wheels respectively. So, the robot is static in pose (x, y, θ) and moves in n_l & n_r counts during a time frame of δt; the new pose of the robot (x', y', θ') is given by calculating the following mobile robot's orientation from the encoder values

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{pmatrix} + \begin{pmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{pmatrix} \tag{4.12}$$

where,

$$R = l/2(nl + nr)/(nr - nl) \tag{4.13}$$

$$\omega\delta t = (nr - nl) \times step/l \tag{4.14}$$

$$ICC = [x - Rsin\theta, y + Rcos\theta] \tag{4.15}$$

The Instantaneous Center of Curvature and other parameters like encoder values are

calculated using this equation. The Kinematic equation depends on geometry and design of the robot. Different equation will come for different designs.

Chapter 5

Robot Modeling in ROS

"With faith, discipline and selfless devotion to duty, there is nothing worthwhile that you cannot achieve."

Muhammad Ali Jinnah

In previous Chapter, we have discussed the mathematical modeling of our differential drive cleaning robot. In this chapter, we will discuss the modeling of robot in ROS. The main use of the robot that we are going to design in this chapter is to clean the environment in homes, hotels, factories and offices etc. The robot is named Cleaning robot. We will cover the complete modeling of this robot in this chapter.

5.1 Robot Model

There are mainly three steps to follow in modeling of robot in ROS. First of all there is need to create 3D model of robot either in solid works or auto-cad. In the second step, there need to make stl mesh files which are integrated in ROS environment. There need too much heed while making robot in degrees of freedom of each part must be created separately. To create a robot, there must be saperate mesh files for wheels and base etc. because base will be static while dynamic parts include wheels. The 3rd step is to create URDF with proper joints and links.

We have created autonomous cleaning robot model.To model the Robot, lets consider its essential components:

- One Chassis
- Two wheels, attached to the chassis
- One front caster wheel, attached to the chassis
- One back caster wheel, attached to the chassis
- A link for laser sensor, attached to the chassis

Modeling of this autonomous robot in ROS includes the following steps:

1. Modeling using URDF
2. Creating Macros
3. Including Macros and Xacro
4. Using Rviz to visualize

In ROS, we represent robot models in an XML format called Unified Robot Description Format (URDF). This format is designed to represent a wide variety of robots, from a two-wheeled toy to a walking humanoid. You can imagine these components forming a tree: the chassis is the root, with connections to each of the rear wheels, the front caster and the back caster. In fact, URDF is only capable of representing robots whose kinematics can be described by a tree; looping structures are not allowed and fortunately are uncommon for robots. We will translate this tree-like narrative description of the Robot into the language of URDF, which is focused primarily on links and joints:

- A **link** is a rigid body, such as a chassis or a wheel.
- A **joint** connects two links, defining how they can move with respect to each other.

5.1.1 Joint Types

When the model components are individually created in solid works then its stl files are compatible with xml format of URDF. All the links are joined with the help of joints. The basic concept of links and joints are given in Figure 5.1. When two links are joint to-gather which is known as parent link and child link. The type of the movement of each element is determined with the help of type of joint b/w the links. Below are the joint

types of robot which can be used the type of movement of each element in the model is determined by the type of joint between the two links [25]. The joint type can be any one from the following:

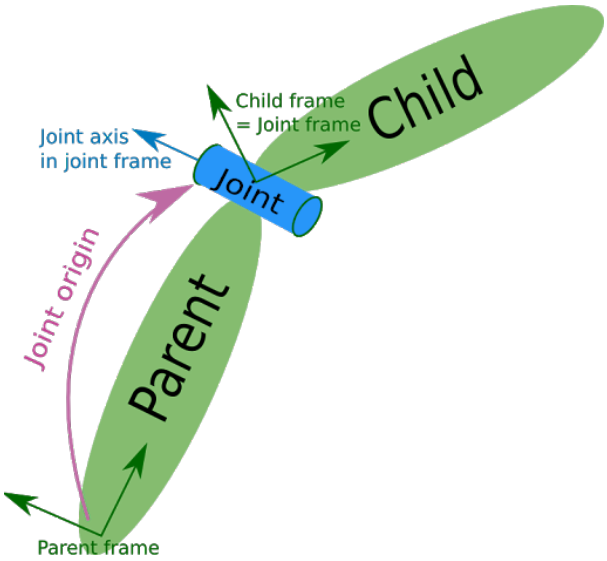


Figure 5.1: Robotics structure using concept of links and joints [26]

Type	Description
Fixed	No DOF, joints are fixed
Revolute	rotation around axis, restricted movement within specified upper & lower limits
Continuous	Resemblance with revolute joints and have no limits of rotation
Prismatic	slides along the axis
Planar	Motion is perpendicular to specified axis
Floating	we cannot say it exactly joint because 6 DOF are free

Table 5.1: Join Types Description

5.1.2 Basic Tags

The URDF file includes descriptions of model. This description have name of the object at first and then mesh files are connected through links with the help of geometry tag. Additional informations about the collisions and visuals are also setting with the help of tags. The material having colors and textures are being selected to make it look beautiful[27]. A more detailed composition urdf-file in Table 5.2

Tags	Description
<robot>	Name of robot, description of robot environment
<link>	Contains the name and visual components, objects are connected via joint tag. Inside <link> tag we can see the visuals, geometry etc.
<origin>	adjustment of initial position
<inerial>	defining the mass of object and offset center
<collision>	due to ROS invisible layer visual objects can collide with other objects so it initiates the collision parameters
<joint>	it is between the two links as shown in Figure 5.1, parent, child and connection type is description types of movement are discussed in Table 5.1

Table 5.2: Basic Tags Description

5.1.3 Solid works 3D model

Below shown is the solid works 3d CAD Model while complete detailed drawing is given in appendix Engineering drawings

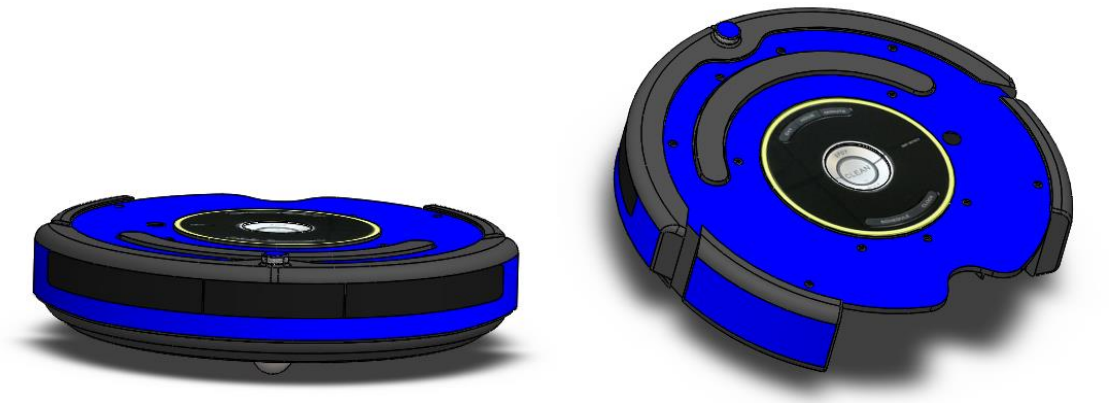


Figure 5.2: Solid Works Model

5.2 URDF Links

So, the URDF allowed us to make the fully functional model. let us discuss the coding in xml format for making of robot.

```
  <robot name="cleaning_robot" xmlns:xacro="http://www.ros.org/wiki/xacro" >
<!-- body -->
<xacro:property name="base_x" value="0.33" />
<xacro:property name="base_y" value="0.33" />
<xacro:property name="M_PI" value="3.1415926535897931" />

<xacro:macro name="cleaningrobot">
  <material name="Green">
    <color rgba="0.0 0.8 0.0 1.0"/>
  </material>
```

5.2.1 Setting-up the Properties

In URDF, first of all there need to start with the robot name and chosing the parameters which we are going to use further in URDF. Below given is the code for the setting of parameter like base x and y positions and value of π is written so that we can use it any time we want [28]. Then the material is selected by using the color rgba.

5.2.2 Chassis Link

First of all, we are going to model chassis_link which is the root link or base link of robot. The URDF code for this link written in cleaningrobot.xacro is shown below in which link, inertia, geometry, collision, visual tags are used to load the STL mesh files to create chassis.

```
    <link name="base_link">
      <inertial>
<mass value="2" />
<origin xyz="0 0 0.0" />
<inertia ixx="0.01" ixy="0.0" ixz="0.0"
         iyy="0.01" iyz="0.0" izz="0.5" />
    </inertial>


      <visual>
```

```
<origin xyz=" 0 0 0.0308" rpy="0 0 0" />
<geometry>
  <mesh filename="package://cleaningrobot_description/meshes/body.stl"/>
</geometry>
      </visual>

      <collision>
<origin xyz="0.0 0.0 0.0308" rpy="0 0 0" />
<geometry>
  <cylinder length="0.0611632" radius="0.164975"/>
</geometry>
      </collision>
    </link>
```

All the properties of this link are defined in <link/> tag. It shows the initial pose of robot. <visual> tag describes the visual dimensions of both parts.

5.2.3 Caster Wheels Links

The URDF code for caster wheels is shown below.

```
    <link name="front_wheel_link">
      <inertial>
<origin xyz="0 0 0" />
<mass value="0.01" />
<inertia ixx="0.001" ixy="0.0" ixz="0.0"
         iyy="0.001" iyz="0.0" izz="0.001" />
      </inertial>
      <visual>
<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
<geometry>
  <sphere radius="0.018" />
</geometry>
      </visual>
      <collision>
<origin xyz="0 0 0" rpy="0 1.5707 1.5707" />
```

As was done in base link here also the visual dimensions have been defined for the both casters. Above is the code for front caster and for rear caster it will also the same but the origin will change. In above code first origin is setted then mass and inertial parameters are defined. In geometry, the caster of sphere is made and visualized using visual tag.

5.2.4 Cliff & Wall Sensors Links

The URDF code for cliff and wall sensors is given below. Cliff sensors are used to avoid the drop off of the robot and wall sensors avoids the bumbing into wall. When the robot is in contact with the wall, this sensor gives the value to controller to invert the direction of velocity. In result, the robot will not bump or drop from stairs or any other height.

```
    <link name="wall_sensor_link">
      <inertial>
<mass value="0.01" />
<origin xyz="0 0 0"/>
<inertia ixx="0.001" ixy="0.0" ixz="0.0"
         iyy="0.001" iyz="0.0" izz="0.001" />
      </inertial>
    </link>
    <link name="left_cliff_sensor_link">
      <inertial>
<mass value="0.01" />
<origin xyz="0 0 0"/>
<inertia ixx="0.001" ixy="0.0" ixz="0.0"
         iyy="0.001" iyz="0.0" izz="0.001" />
      </inertial>
    </link>
```

In the same way other cliff and wall sensors are adjusted in the right, left, front and back side of robot. The cliff sensors are adjusted in the back and front of the robot.

5.2.5 IMU Link

Inertial Measurement Unit (IMU) is used to get the orientation of robot. We can retrieve quaternion orientation, angular velocity and linear acceleration from this IMU sensor. Below given is the URDF code for IMU link.

```
<link name="imu_link"/>
<joint name="imu_joint" type="fixed">
  <parent link="base_link"/>
  <child link="imu_link"/>
  <origin xyz="-0.032 0 0.068" rpy="0 0 0"/>
</joint>
```

5.3 URDF Joints

It is necessary to add joints with the link to make a stable robot. the basic code for the joint is given below

```
<joint name="front_castor_joint" type="fixed">
  <origin xyz="0.13 0 0.0" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="front_wheel_link"/>
  <axis xyz="0 1 0"/>
</joint>
```

There are mainly 4 components in URDF Joints

1. Origin: To set up the origin of joint
2. Parent: The parent link for the joint
3. Child: The child link for the joint like front wheel link in above code
4. Axis: The axis of the joint on which is consists

In this cleaning robot there are joints for each link the following joints are used

- base footprint joint
- left wheel joint
- right wheel joint
- front wheel joint
- rear wheel joint
- imu joint
- base wall sensor joints
- cliff sensors joints
- hokuyo joint for laser scan

5.4 Gazebo Plugins Configuration

The gazebo plugins are used to control the properties of world, sensors. The parameters like joint state publishers subscribers etc. The controllers like differential drive controller are available to interface with gazebo. The codes for plugin discussed below are given in Appendix 1. Below are the plugins used in this autonomous robot [27]

5.4.1 Lidar Laser Scanner Plugin

The lidar plugin is used to interface laser scanner with gazebo environment. With the help of this plugin controller gathers the range data from a simulated array sensor. This plugin includes the ray count, laser count starting from 1, update rate is set to 5. This Lidar can occupy the 360 degrees having resolution is setted to 1 and min and max ranges are from 0.120 to 3.5 to detect the obstacle and walls. The topic for this laser scan is "scan" and frame name is hokuyo_link.

5.4.2 Differential Drive Controller

As our robot is controlled by differential drive so we have used the differential drive plugin to control its simulation in gazebo. It provides the control to drive system. In this plugin

we have to set up the links and frames. Starting from plugin name topic of cmd_vel, odometry with odom topic. The data is published by using publishOdomTF [27]. The base_footprint is selected as base frame. The publishers are made for wheels and their joints. Wheels separations, wheel diameters, wheel acceleration and wheel torque are 0.26, 0.66, 1, 10 respectively which are properly defined.

5.4.3 IMU Plugin

This sensor simulates the inertial measurements. It provides the orientation by using offsets of x, y, z and roll, pitch, yaw.

5.4.4 Joint State Publisher

This controller is used to publish the joint state values given in URDF to transform for all the joints. This will provide the interaction with each and every joint.

5.5 Sensors Integration

5.5.1 LiDAR Sensor

Lidar is a sensor which use laser light to illuminate the target and measure the back reflection to find the measuring distances (ranges). This instrument shown in Figure 5.3 fires laser light on the surface of target with high pulses of about 150000 per second. It is used in generating the 2D maps using the Gmapping techniques in ROS environment.Using this sensors interfacing with RVIZ and it will communicate this data to Raspberry Pi, Arduino and ROS will start to generate a map. After saving map of any place, the robot has all the data in that work-space.

5.5.2 IMU Sensor

This sensor is changign the electronic and industrial market day by day. This device gives the combination of accelerometers, gyroscopes and some of them gives magnetometers also to measure specific force, angular rates and mainly the orientation of body. Below in Figure 5.4 IMU sesnsor is shown

Figure 5.3: Light detection and ranging sensor for distance measurement [29]

Figure 5.4: Inertia Measuring Unit (IMU) Module [30]

5.6 Motion Integration

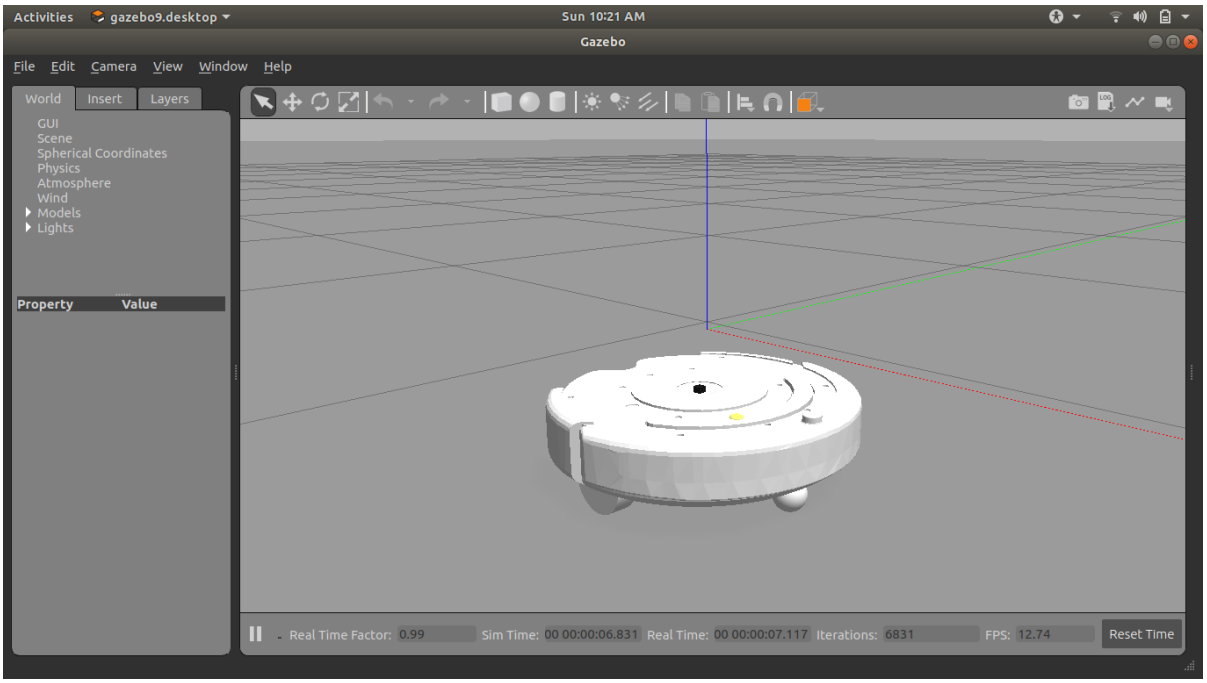
Now were ready to Simulate our Robot model in Gazebo. There are a few different ways to do this. Because we want to use some ROS tools with our simulated robot (as opposed to working solely within Gazebo), well follow this pattern, using roslaunch to automate things:

1. Load the robots URDF model into the parameter server.
2. Launch Gazebo (e.g., with an empty world).
3. Use a ROS service call to spawn an instance of the robot in Gazebo, reading the URDF data from the parameter server.

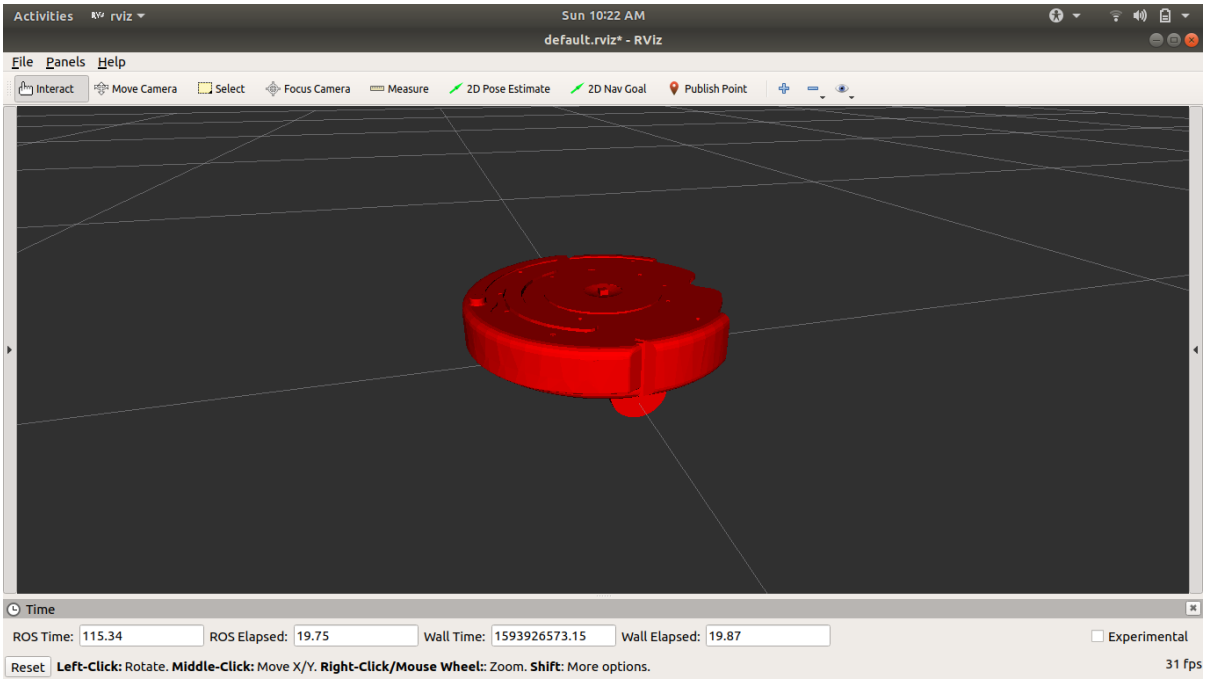
This process might seem a little roundabout, but its actually a very flexible way of doing things. For a start, it gets the URDF model onto the parameter server, where it can be accessed by other nodes. By convention, the URDF model is stored in the

parameter server under the name /robot_description (you can use another name for this parameter, but then youd have to change the default settings for many tools). Once its on the parameter server, the URDF model can be used by tools like rviz, which needs the model to visualize the robot, or a path planner, which needs the model to know the robots shape and size.

5.6.1 Gazebo Simulation

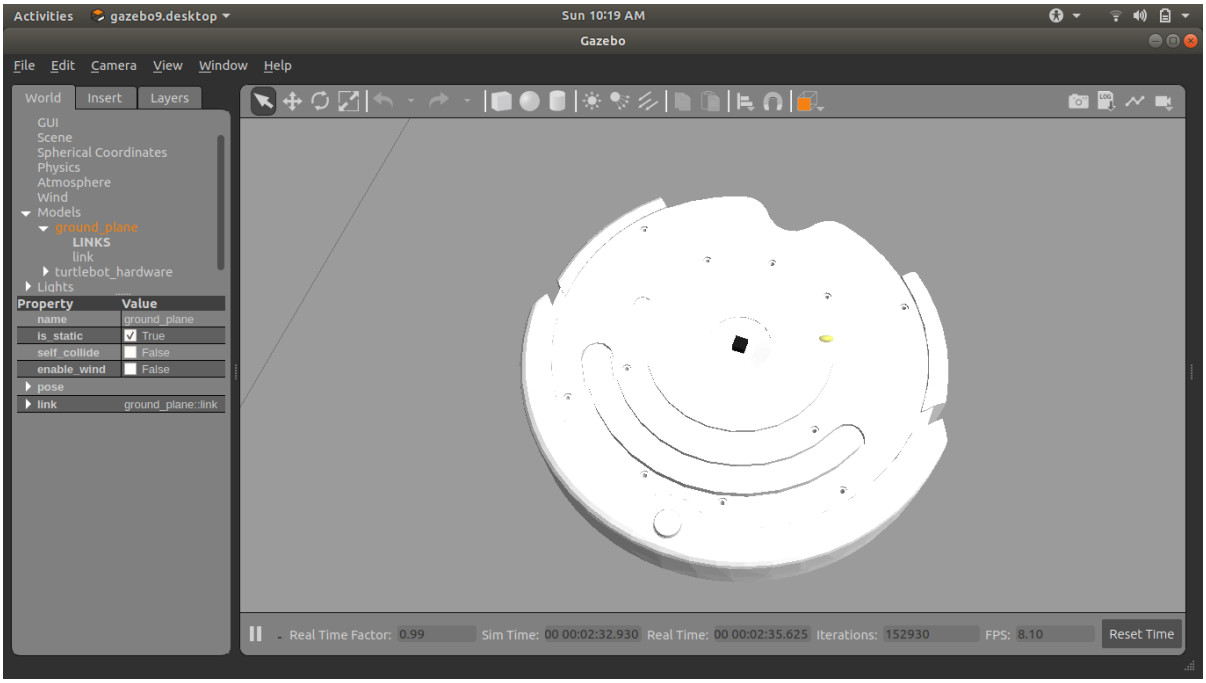


(a) Robot URDF Isometric View in Gazebo

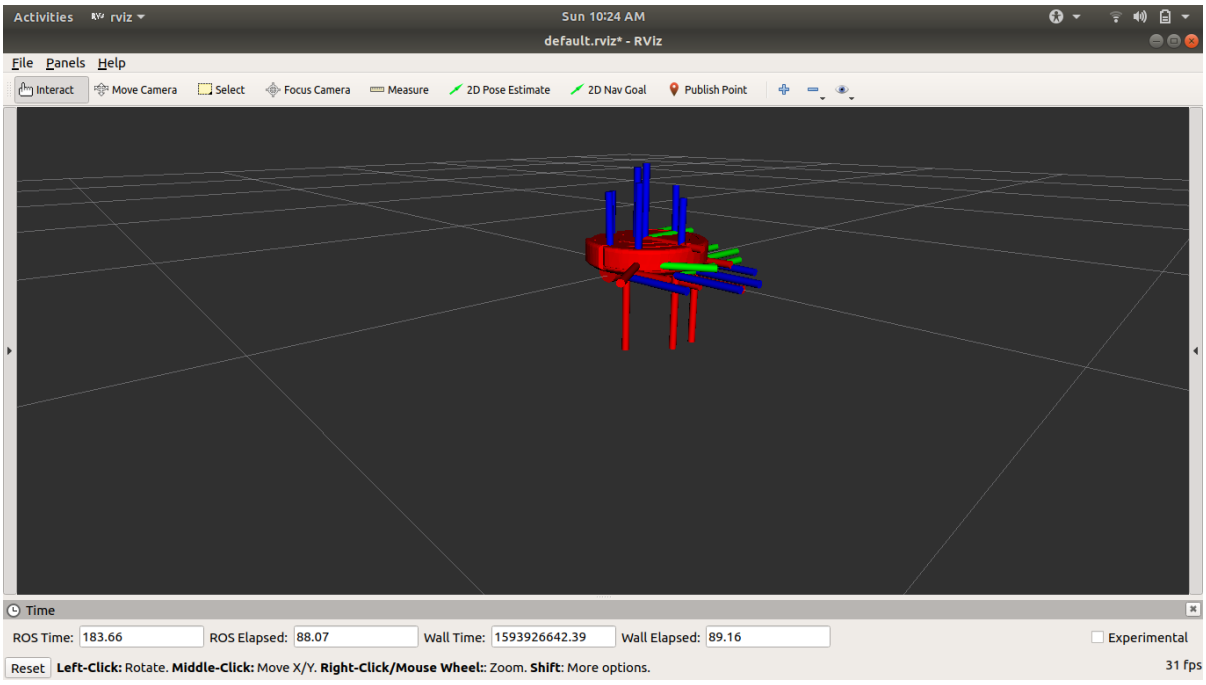


(b) Robot URDF Isometric View in Rviz

Figure 5.5: Robot URDF Model



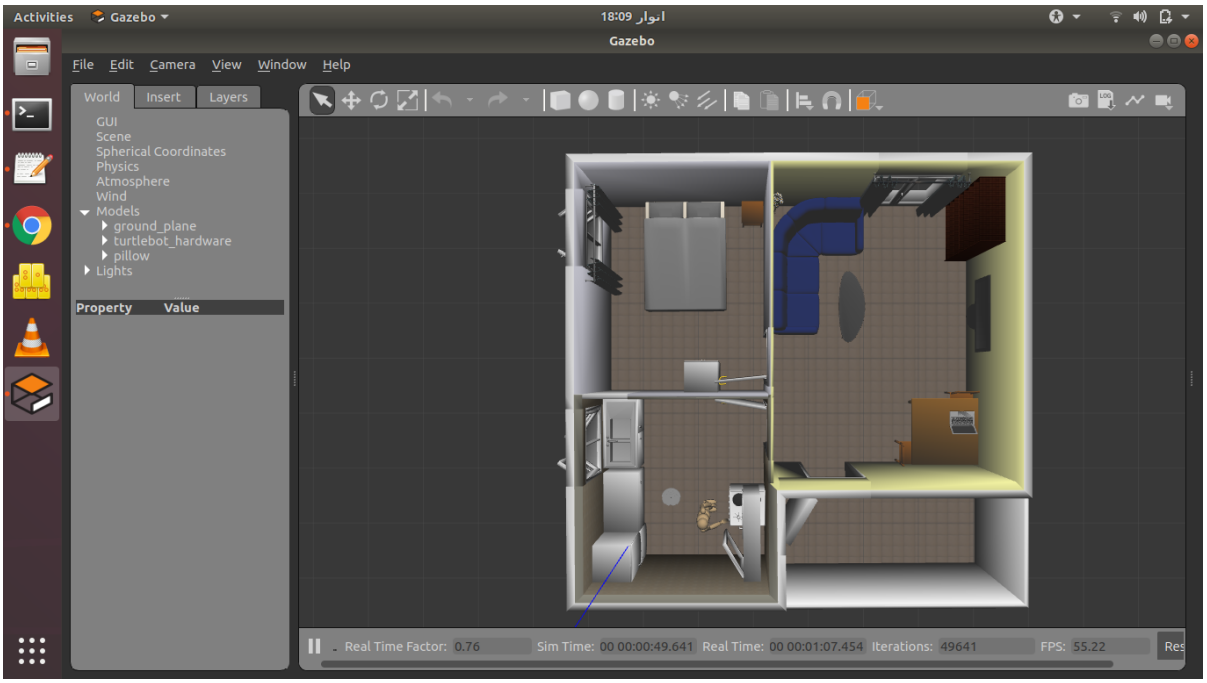
(a) Robot URDF Top View



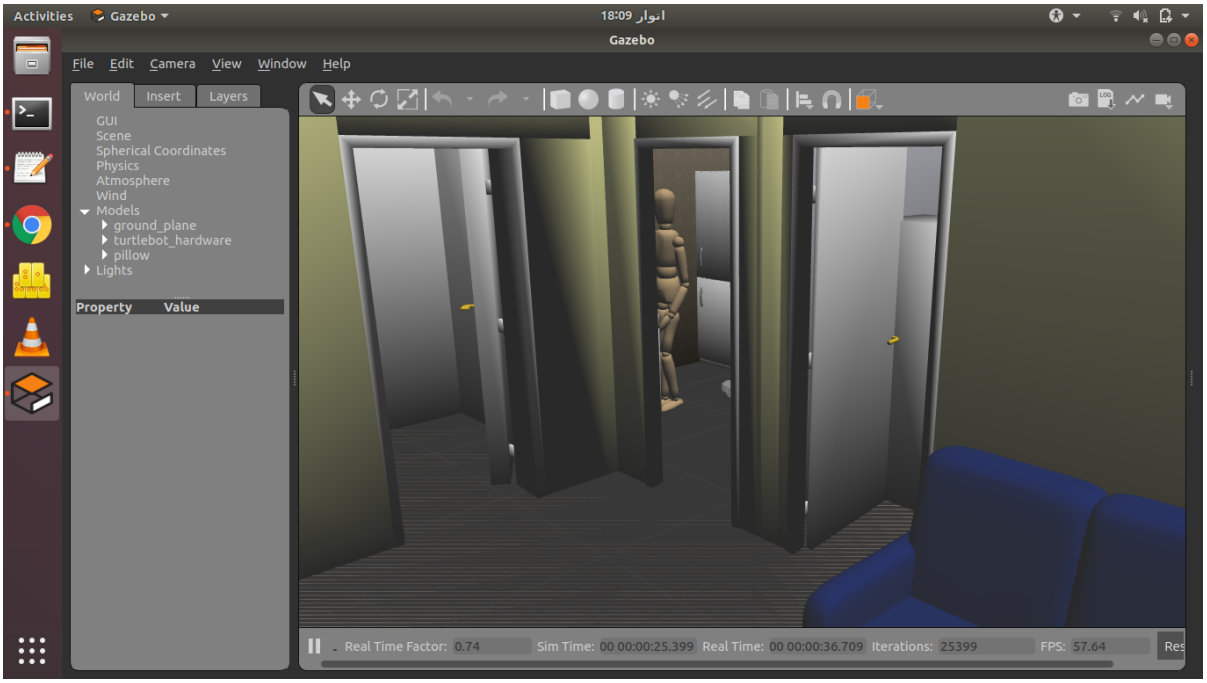
(b) Robot URDF Transformations

Figure 5.6: Robot URDF Model

5.6.2 Gazebo Environment/World



(a) Gazebo world Top View



(b) Gazebo World Inside View

Figure 5.7: Gazebo World/Environment of Home

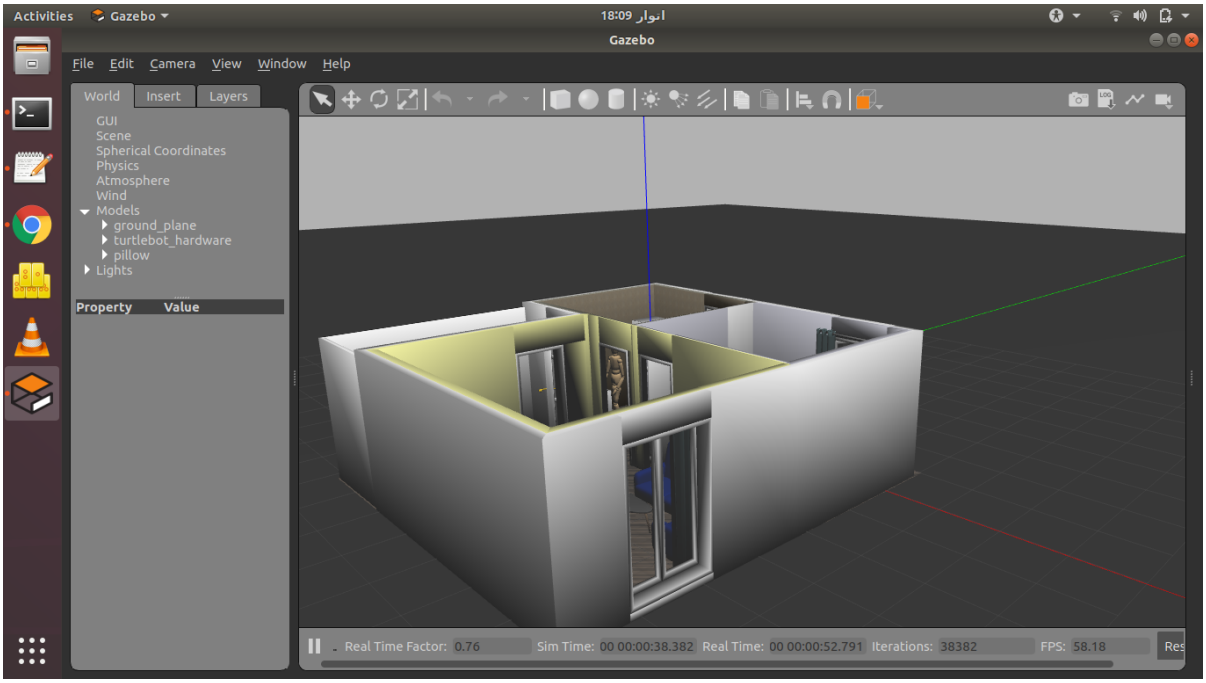


Figure 5.8: Gazebo World Isometric View

Chapter 6

Simultaneous Localization & Mapping (SLAM)

"It is only by learning from mistakes that progress is made."

Sir James Dyson

In this chapter, we will take a look on simultaneous localization and mapping (SLAM). The basic purpose of SLAM is to build map and localize the robot on the same time. The robot can be run by two ways to make a map either the robot moves autonomously or controlled manually to move a robot for map making. The main purpose is to make map correctly.

The basic way used to localize the robot is use of odometry. In a robot, odometry consists of combination of the displacement measured with the help of encoders for the estimation of orientation and position. The involve the odometry error in this way. SLAM works for the correct estimation of orientation and mapping. That is why, this method is very difficult because of errors of both the robot and map states affect each other.

6.1 Mapping

In order to navigate in any environment, we need a few key elements. For a robot this is no different. What we need is:

- A map of the environment

- A route to take (path planning)
- The current location on the map (localization)
- A route to take (path planning)
- Obstacle avoidance

A map is mandatory for navigation. There are two ways to get a map:

- Use a pre-existing map
- Build by the robot itself (mapping process)

The mapping process is called SLAM (Simultaneous Localization And Mapping). Here we are using a specific type of SLAM called G-mapping. Using the Laser Scanner, the Robot can get information that are the obstacles in the environment. This information is then processed by the slam-gmapping node. After that it published to map topic. By using the map_server, map_saver and slam_gmappig node the map created of the world is shown below in Figure 6.1

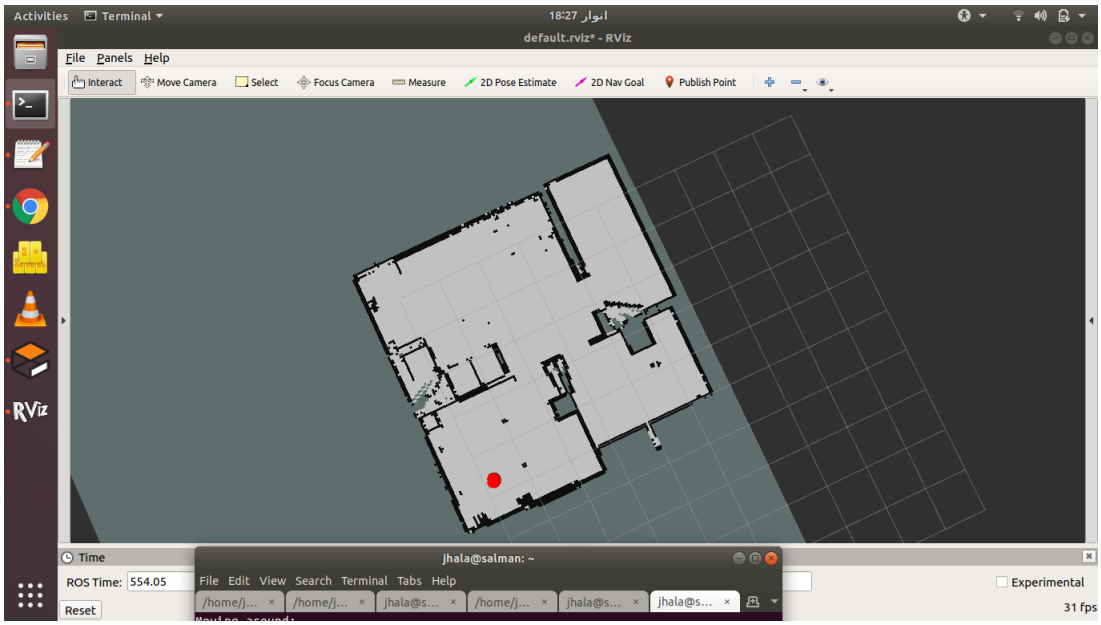


Figure 6.1: Map of World

6.2 Localization

Localization methods are grouped into two categories: Global and Local.

6.2.1 Global

- Give a location with respect to the world
- Often inaccurate compared to local methods

6.2.2 Local

- Give a location with respect to local sensor feedback
- Can be highly accurate compared to global methods
- For example, laser scanner and cameras

In general, Local Methods of localization are more accurate. Sometimes global localization methods cannot give accurate enough results to localize a robot in a map. Local sensor readings can give accurate positioning with respect to obstacles in the map, but there may be multiple places on the map that look the same. Location can be narrowed down by estimating a range of positions a robot is likely to be by tracking how much the robot moves and by taking many measurements [31].

6.2.3 AMCL

The amcl stands for Adaptive Monte Carlo Localization. This is a probabilistic localization system for a robot moving in 2D which uses a particle filter to track the pose of a robot against a known map. The known map in this case is the one we just created. The image below shows the TurtleBot in the map that has no been yet know that where it is exactly in the map.

A python script is used for the localization of the robot. below given is the figure of that python script and localization in Rviz.

```python
import rospy
from geometry_msgs.msg import Twist
rospy.init_node('localzlization')
pub = rospy.Publisher('/cmd_vel',Twist, queue_size = 10)

PI = 3.1415926535897
def rotate(vel_msg,pub):
        angular_speed = .5
        vel_msg.angular.z = angular_speed
        t0      = rospy.Time.now().to_sec()
        angle_travelled = 0

        while ( angle_travelled < PI/2.0 ):
                pub.publish(vel_msg)
                t1 = rospy.Time.now().to_sec()
                angle_travelled = angular_speed*(t1-t0)

        vel_msg.angular.z = 0
        pub.publish(vel_msg)


print "let us localize robot"
pub = rospy.Publisher('/cmd_vel',Twist, queue_size = 10)
vel_msg = Twist()
rotations = 1
current_rotation = 0
while current_rotation < rotations:
        rotate(vel_msg,pub)
        current_rotation+=0.25
```

Figure 6.2: Python Script For Localization

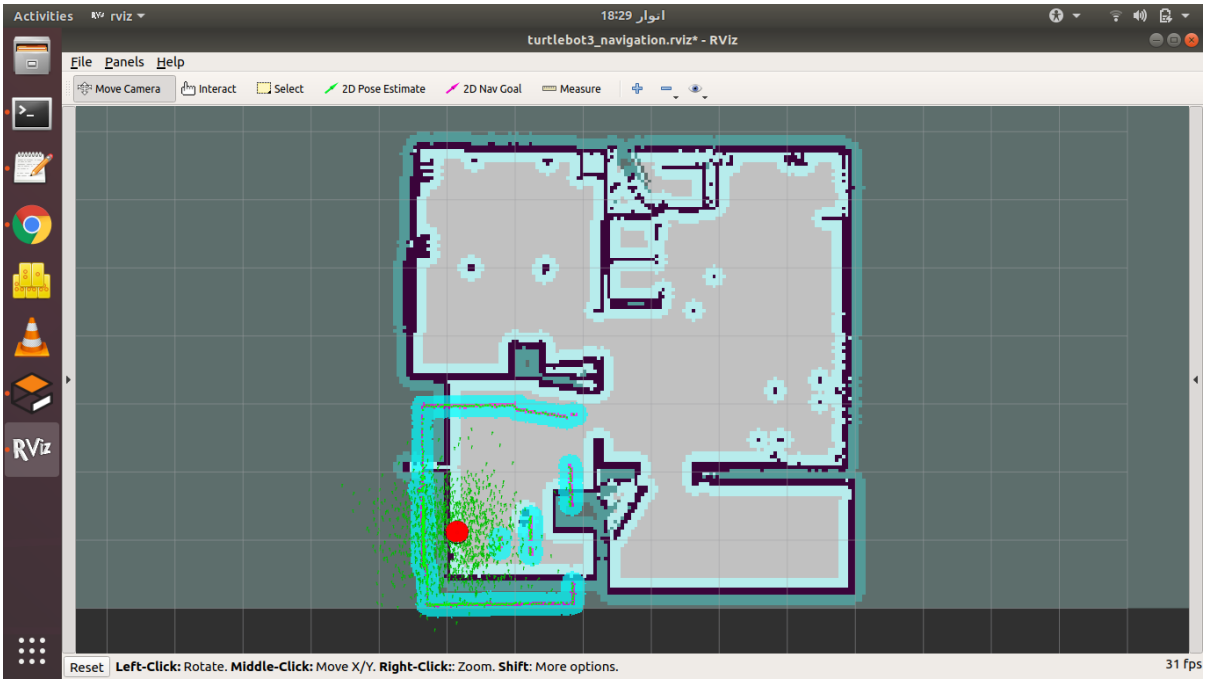


Figure 6.3: Localization in world

In Figure 6.3 The round black circle is the cleaning robot. It's surrounded by green arrows which are probabilistic estimates of its actual position. Then you see the edges from the square as in our original map. One of its edges is purple with (and a thin white line form the laser scan is on it) [32]. This is one of 'measurements' the robot has taken to localize itself. So now we have to move the robot and let it take more measurements to improve localization by running the python script shown in Figure 6.2.

Now moving the robot around using the given python script above and observe the green arrows around it. Since we get more certainty about where the robot is, the arrows should come close and closer indicating more a more accurate probabilistic model. It should be similar to the following

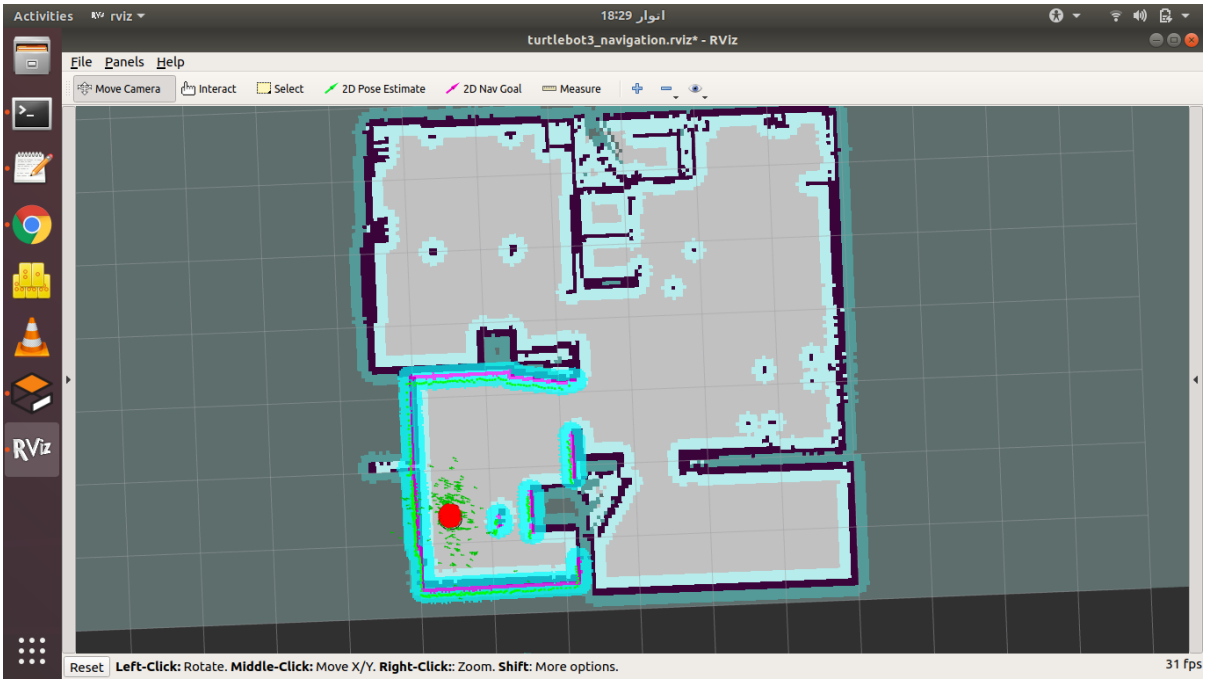


Figure 6.4: Accurate Probabilistic Model of Map

6.3 Autonomous Mapping Using Frontier Exploration

The process which is used to select target points which gives the maximum contribution to gain function for an initially unknown environments. The region between explored and un-explored space is known are frontiers regions. In this section we will discuss

the autonomous frontiers exploration strategy for the autonomous mapping of unknown environment[33].

Exploration is one the fundamental problems for the unknown environment in the field of autonomous mobile robotics, It deals with the study of autonomous creation of map of unknown areas at the same time. This map will be used in navigation tasks for obstacle avoidance [34]. There are many advantages and application in field of space robotics, sensors deployment and military robots.

Further more the map generated with the help of this technique is validated by the map generated manually with the help of teleop.

The main aim of Algorithm is to find frontiers and with the help of G-mapping, implement the SLAM. The terminologies which are used in this section are given below

- **Unknown Regions** Region which is not covered by robot sensors.
- **Known Region** Region which is already covered by robot sensors.
- **Open Space** Region which contains obstacles and known to the robot.
- **Occupied-Space** Region With black covered in map known as region with obstacles.
- **Occupancy Grid** When the environment is represented by grid. Each and every cell will represent either it is 0 or 1. If one means covered area with obstacles.
- **Frontiers** These are the segments which are separated by known and unknown regions

The terminologies given above are represented in the form of Figure 6.5

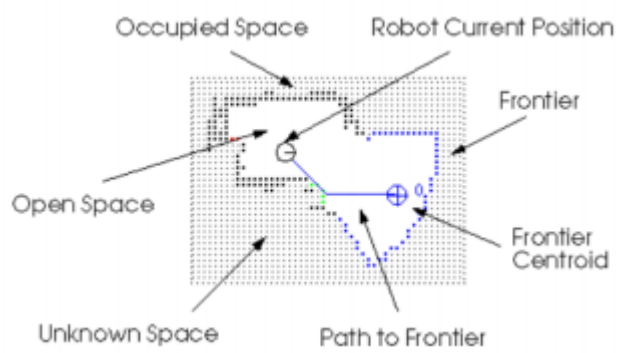


Figure 6.5: Frontiers Representation [35]

6.3.1 Working of Algorithm

The whole algorithm is based on the breadth first search called BFS [35]. Below are the steps to run the frontiers based algorithm.

1. After setting up the robot, the BFS search is made run on the whole grid and frontiers are added to the data in queue structure
2. In 2nd Step, frontiers are again run on the entire grid to get the final frontiers.
3. In 3rd Step, There need to find the centroids of the frontiers. which are queue in another data structur which are arranged in the form of of increasing order of their euclidean distances. so that our robot can move in the closer frontiers.
4. In 4th Step, The robot reach the frontier and perform 360 degrees of rotation to create a map of the area.
5. These frontiers are continuously explored until the whole region is explored and no more frontiers left which means that whole environment is now mapped.

6.3.2 Possible Indicators

The key feature is to reduce the time complexity factor of the algorithm. There are four indicators to ensure that when the algorithm is called only the known regions are scanned. It is known by Frontiers Based Algorithm (FBA)

1. **Map Open List** List of points arranged by the outer BFS
2. **Map Close List** List of points disarranged by outer BFS
3. **Frontier Open List** List of points arranged by inner BFS
4. **Frontier Close List** List of points disarranged by inner BFS

There four lists are used by map data structure to implement algorithm [**sl4**]. And also these lists ensure that the frontier points is not assigned to the multiple frontier points.

Algorithm 1 Frontier Based Algorithm For Autonomous Mapping

Result: MAP

Inputs:

$queue_m$ // queue, used for detecting frontier points from a given map

$queue_f$ // queue, used for the extracting a frontier from a given frontier cell

pose // current global position of the robot

$queue_m \leftarrow \emptyset$

ENQUEUE($queue_m, pose$)

mark *pose* as "map-open-list"

while $queue_m$ *is not empty* **do**

  $p \leftarrow DEQUEUE(queue_m)$

  if *p is marked as "map-close-list"* **then**

    continue

  end

  if *p is a frontier point* **then**

    $queue_f \leftarrow \emptyset$

    $NewFrontier \leftarrow \emptyset$

    $ENQUEUE(queue_f, p)$

    mark p as "frontier-open-list"

  end

  while $queue_f$ *is not empty* **do**

    $q \leftarrow DEQUEUE(queue_f)$

    if *q is marked as {"map-close-list","frontier-close-list"}* **then**

      continue

    end

    if *q is a frontier point* **then**

      add q to NewFrontier

      forall $\omega \in adj(q)$ **do**

        if ω *not marked as {"frontier-open-list", "frontier-close-list"}* **then**

          ENQUEUE($queue_f, \omega$)

          mark ω as "frontier-open-list"

        end

      end

      mark q as "frontier-close-list"

    end

  end

  save data of NewFrontier

  mark all points of NewFrontier as "map-close-list"

  forall $v \in adj(p)$ **do**

    if *v not marked as {"map-open-list", "map-close-list"}* ***and*** *v has at least one "map-open-space" neighbor* **then**

      mark v as "map-open-list"

    end

  end

  mark p as "map-close-list"

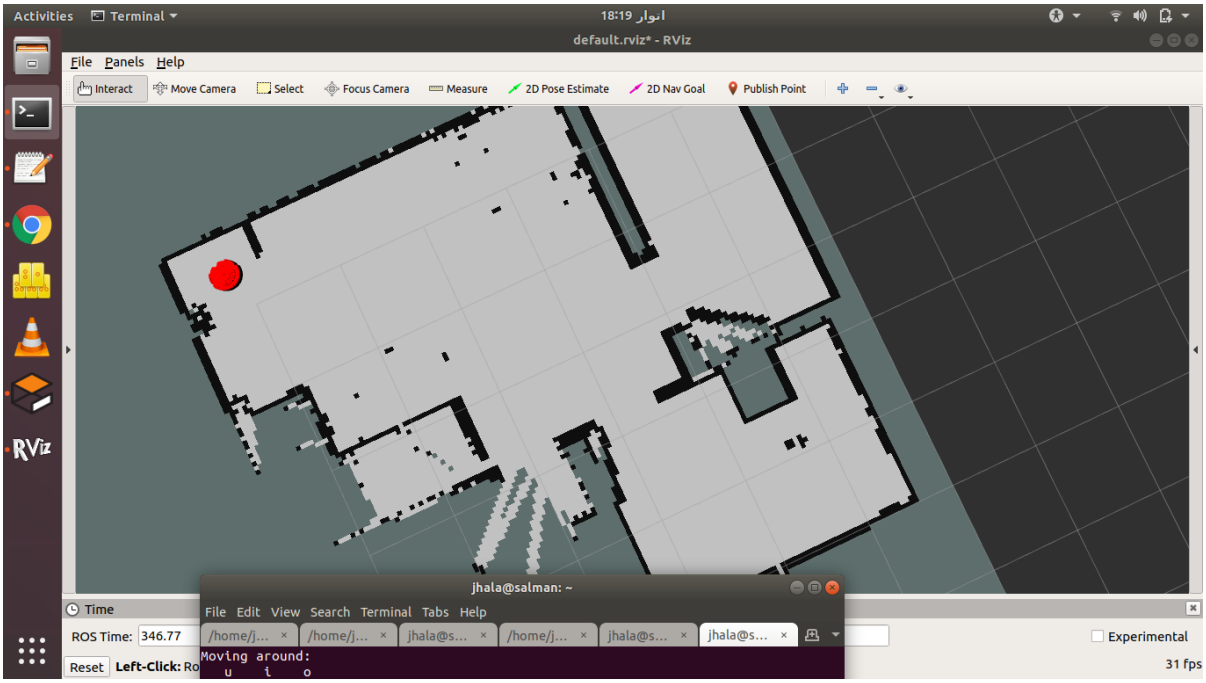
end

6.3.3 Algorithm

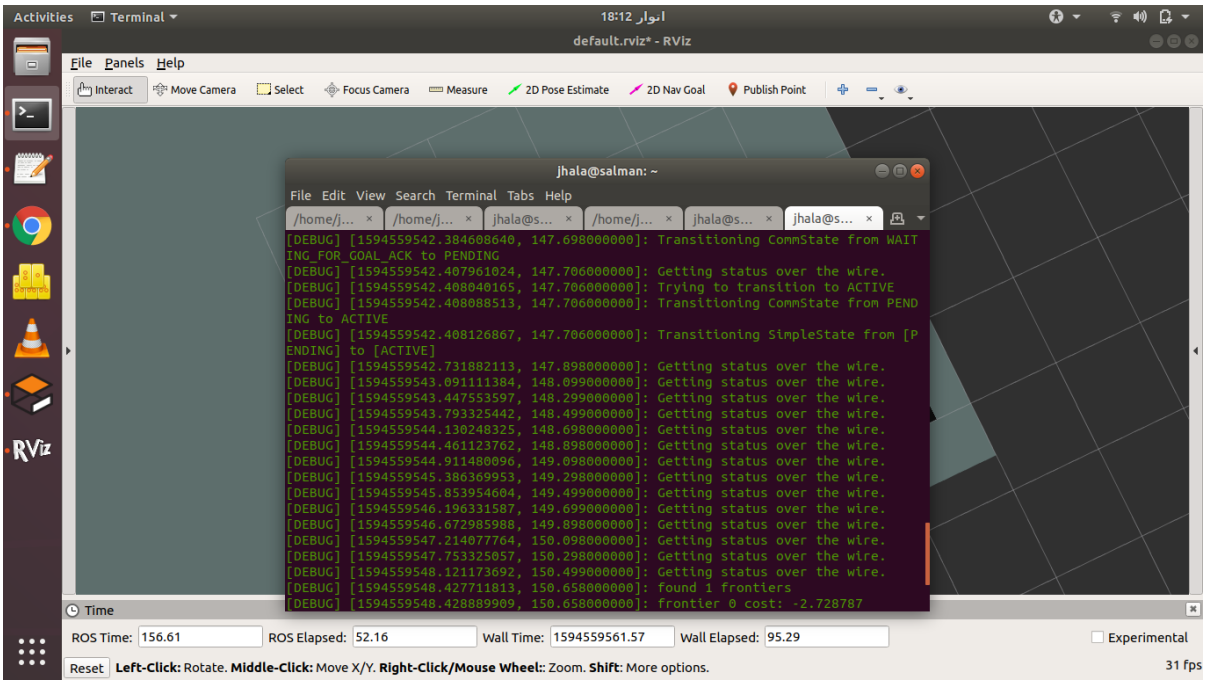
First of all data structure is initialized for queue to find out the points for the occupancy grid. These are known by $queue_m$: Point enqueued. This list is marked by map-open-list. The points which are not included in map-open-list are considered to by in $queue_m$. In addition, each and every point added in this queue have at least one open space neighbour. Due to this only known regions are scanned.

Toward the beginning, the point in occupancy grid relating to the current pose and orientation of the robot is enqueued into an arranged data-structure. Next, a BFS is performed until a frontiers points are experienced. Once, such a point is experienced, another queue is instated to separate its frontiers. Suppose this point is known as $queue_f$. The points which are not part of map-open-list are arranged or enqueued in this queue. The points of frontiers which are obtained from outer BFS are enqueued. Then a new BFS is used to find the connected frontiers points.

All focused points being visited in the inward BFS are set apart as frontier-Open-List. At whatever point an outskirts point is experienced in this BFS, it is put away in a rundown containing all other frontiers focused points regarding their separate frontiers. These focused points are set apart as Frontier-Closed-List. A frontier point is summed up with a frontier list just in the event that it's anything but a piece of the Frontier-Open-List, Wilderness Close-List or Map-Closed-List. This guarantees that every frontier point is alloted to a one of a unique frontiers [34, 36]. Utilizing this filtering strategy, it is ensured that all frontiers focused points will be appointed to a one of a unique frontiers toward the finish of every run of algorithm.



(a) Exploring Environment



(b) Exploration Working

Figure 6.6: Mapping Through Frontier Exploration

Chapter 7

Autonomous Navigation

"Vision without power does bring moral elevation but cannot give a lasting culture."

Allama M. Iqbal

7.1 Introduction

In this chapter, we are going to discuss on autonomous navigation. In Navigation tasks, vehicle itself is able to plan its path and execute that plan without any human intervention. This path is totally based on the input coming from different sensors and also map of environment which is already created. This chapter relates with the motion planning and dealing with partial observably of the environment. The autonomous navigation problem is now solved using different algorithms [37]. In contrast with this, dynamic and unknown environments still represent a challanging task due to partial observable environment. Dynamic environment consists of different objects that can change their shape, position and orientation for example doors, humans, animals etc. There are 3 possibilities of assumptions that navigation system considered when it plans movement [38].

1. **Known Space Assumptions** In this assumption whole area or region is supposed to be known to robot via map which is already generated [39, 40]

2. **Free Space Assumption** In this assumption the unobserved space is supposed to be without any of obstacle. It often without previous map. [37, 41]

3. **Unknown Space Assumption** In this assumption, the environment is totally observable. it is used for autonomous map-building tasks, not for navigation tasks. [42]

Most of the navigation systems have known space assumption, the environment is static in most of cases or even small dynamic obstacles like humans exists in case of cleaning robot purposes and map doesn't have any errors.

7.2 A* Algorithm

A* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.) [43]. It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied. Pseudocode The following pseudocode describes the A* algorithm:

7.2.1 Algorithm

7.2.2 A* Example

An example is shown in Figure 7.1 of an A* algorithm in action where nodes are cities connected with roads and h(x) is the straight-line distance to target point

Algorithm 2 A* Algorithm

Input: Graph, start, goal
Output: path
Function `A_Star`$(start, goal)$**:**

```
Set all nodes g(s) = ∞
start = 0
open.insert(start, g(s)) + h(start, goal)
PATH(start)= NONE
while open is not empty OR s is not goal do
    s = open.Pop()
    forall s' ∈ successor(s) do
        if g(s) + cost(s, s') < g(s') then
            g(s) = g(s) + cost(s, s')
            open.insert(s', g(s') + h(s', goal))
            PATH(s') = s
        end
    end
end
```

End Function

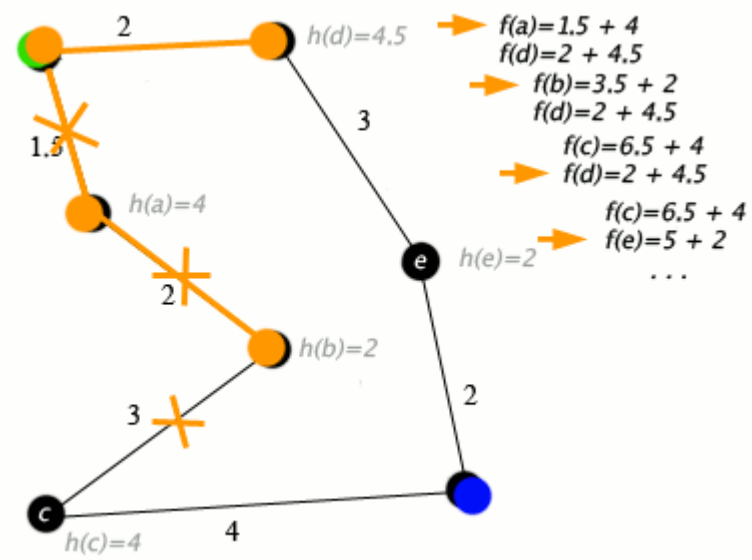


Figure 7.1: A* Example [44]

7.2.3 A* MATLAB Animation

Below given are the demo animations made in MATLAB. The goals and start positions are automatically generated and then the algorithm start working in Figure 7.2 gives the Demo Animation of A* Before Path Estimation and Figure 7.3 gives the Demo Animation of A* After Path Estimation

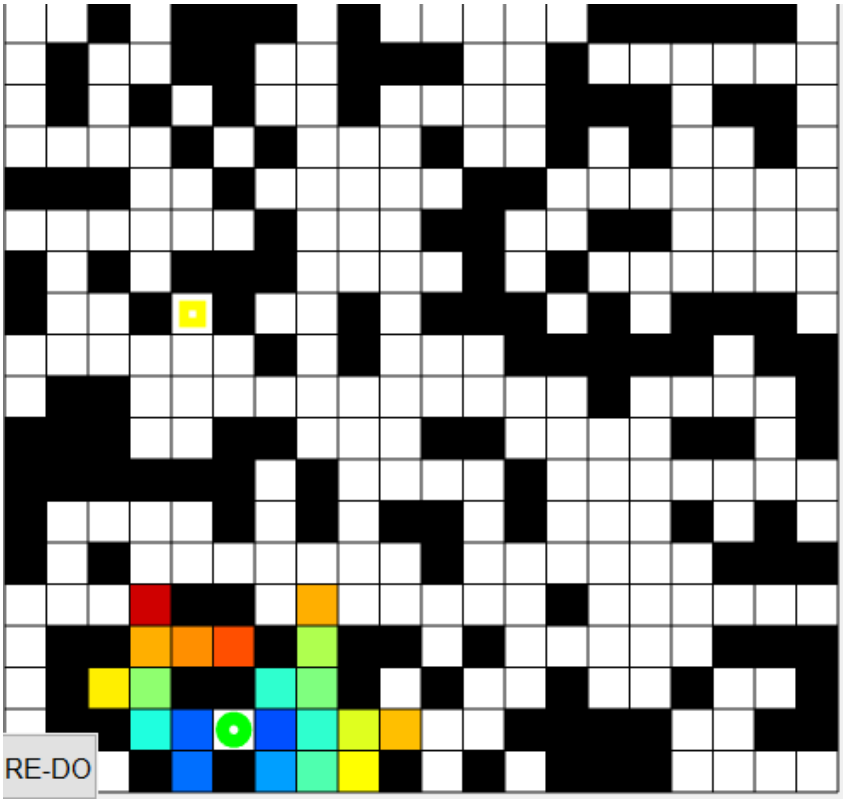


Figure 7.2: Demo Animation of A* Before Path Estimation [45]

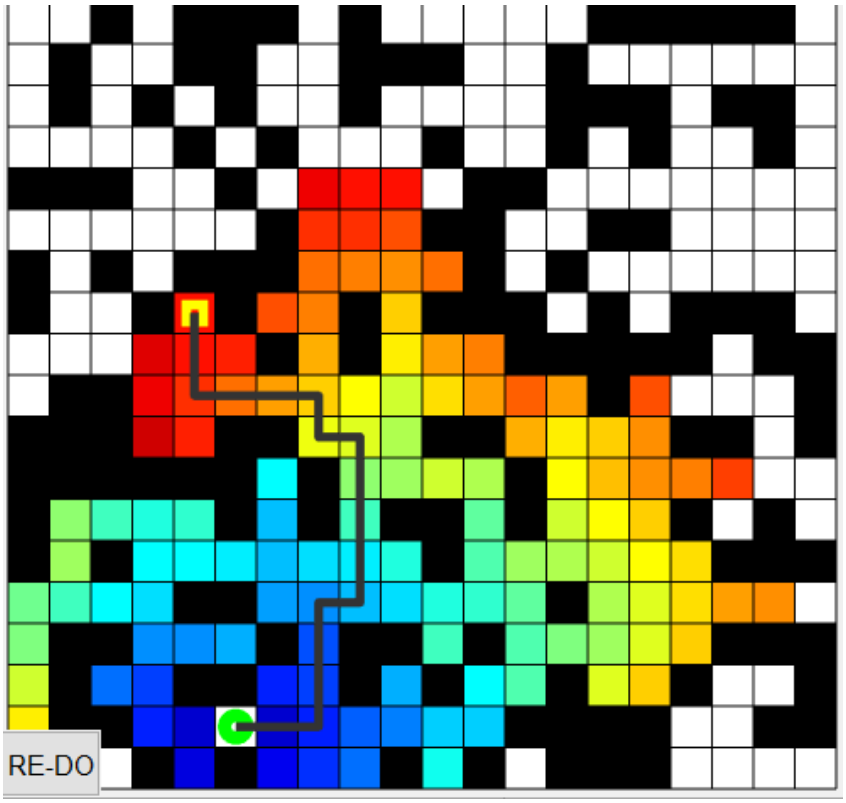


Figure 7.3: Demo Animation of A* After Path Estimation [45]

7.3 ROS Navigation Stack

For the creation of 2D maps with the help of lidar sensor to scan the environment with a differential drive robot ROS navigation stack is compatible. IMU sensor is used to check the odometry and pose of robot. A goal pose is used in it for the movement of robot. The node move_base is mainly used in navigation stack. For the accomplishment of goal local and global planners are used for the management of communication with in navigation stack. A node gathers the information coming from different sensors and put this information in transformation node. After this with the help of odom node starting and goal position can be estimated on the pose of robot [46]. The trajectory will be calculated with the help of information provided from different nodes on move_base and the velocity commands are passed on with the help of base controller. Right now, we are ready to go. Use the 2D post estimate button on the top of your screen to localize the robot., use the 2D Nav goal to give the robot a target position to move to. Then, watch the Cleaning robot as it generates a path and tries to follow it

7.3.1 Navigation Stack Architecture

The following Figure 7.4 shows the architecture of the navigation stack of ROS

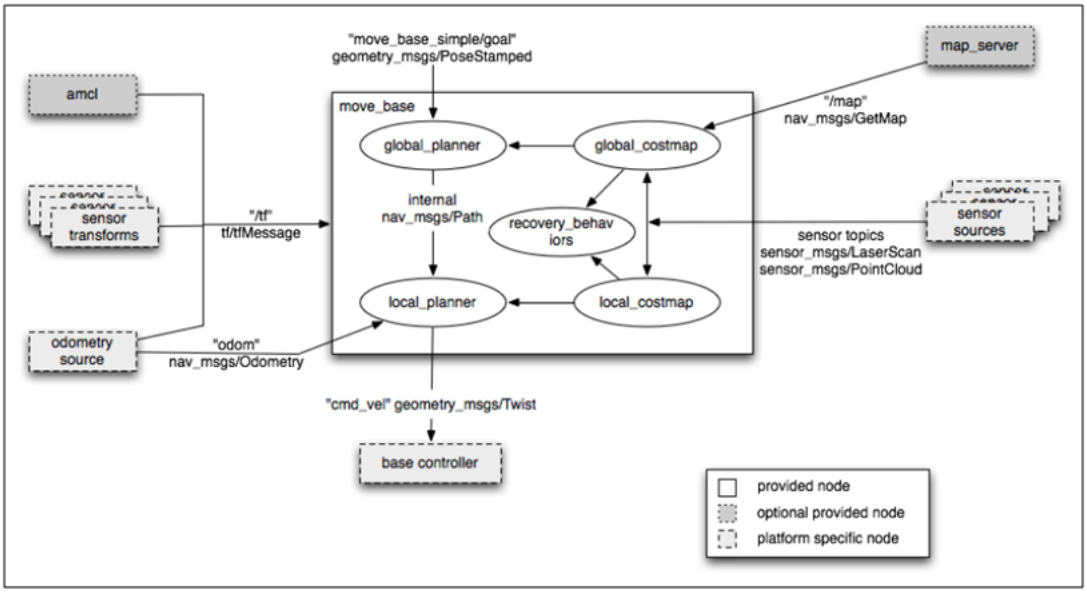


Figure 7.4: Architecture of Navigation Stack [47]

Sensors are providing the input values: sensor sources and odometry source. Based on

those two data sources, the algorithm calculates the sensor transformation, the pose estimate (amcl) and map the environment (map_server). Thanks to all this data, the path planner can take decisions. Sensors are essential, and the LiDAR is the best sensor to get LaserScan or PointCloud data.

Below is the explanation of blocks used in above architecture diagram of navigation stack.

- **Odometry Source** This contains the robot position and orientation data. Main sources includes wheel encoders, IMU and laser scanner i.e. visual odometry. This odometry data is published on navigation stach with odom node and contains a message type of nav_msgs/Odometry.
- **Sensor Source** To map the environment, there is need of laser scan data. These two data inputs combines to make the local and global cost maps [46]. The sensor used here is lidar. The data is in type of sensor_msgs/LaserScan
- **Sensor Transform/tf** The robot coordinate frames are published by ROS tf node.
- **base_controller** This base controller conert the twist output messages into motor velocities. The other nodes includes the amcl and map server to allow the localization of the environment [48].

7.3.2 Understanding *move_base* Node

The package move_base has a node called move_base node. The main objective of node is to control the robot and move it from start to goal position with the help of navigation stack nodes. The *move_base* connects both the local and global planners to decide the path trajectory to the goal. For obstacle avoidance it connects the global costmap to local costmap.

The move_base node basically is an implementation of SimpleActionServer which takes a goal pose with message type (geometry_msgs/PoseStamped). We can send a goal position to this node using a SimpleActionClient node. The move_base node subscribes the goal from a topic called move_base_simple/ goal , which is the input of the Navigation stack, as shown in the previous diagram.When this node receives a goal pose, it links to components such as global_planner , local_planner , recovery_behavior ,

global_costmap , and local_costmap , generates the output which is the command velocity (geometry_msgs/Twist), and sends to the base controller for moving the robot for achieving the goal pose

7.3.3 Functionalities of Navigation Stack

Below is the list of packages which are linked to this node

- global_planner
- local_planner
- ratate-recover
- clear-costmap-recovery
- costmap-2D
- AMCL
- map-server
- Gmapping

Below are the functionalities of the ROS navigation stack. When the robot is properly publishing the odom data, tf information and sensor data comming from laser scanner. Already have a map and base controller than we are ready to work with navigation stack.

- localizing on the map
- Sending a goal and path planning
- Collision recovery behaviour
- Sending command velocity

7.3.4 Map Tagging

After autonomous navigation there is need to know where the robot is. There are four parts in the map room, kitchen, launch and corridor. So, we have tagged the map and if we type kitchen the robot will move to kitchen. Figure 7.5 shows the tags on map.

Figure 7.5: Map Tagging

7.4 Implementation Simulation of Autonomous Navigation

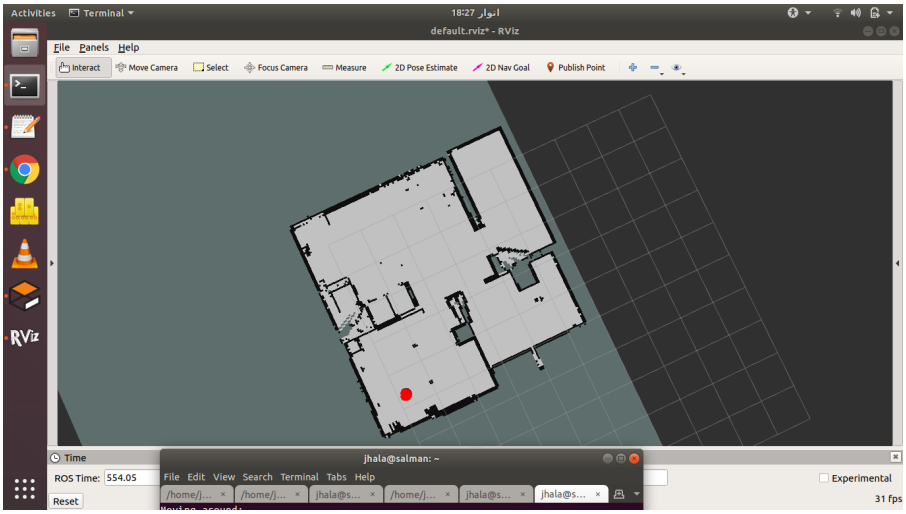


Figure 7.6: Initial world Map in Rviz

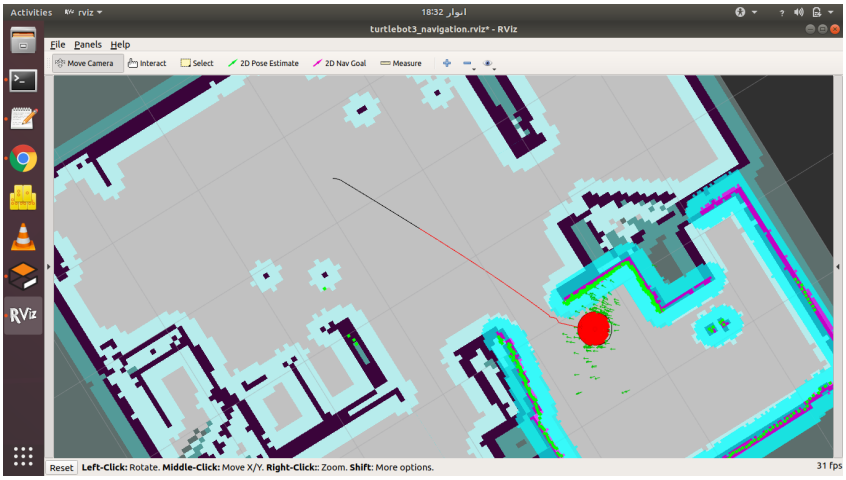


Figure 7.7: Path Planning From Start to Goal State

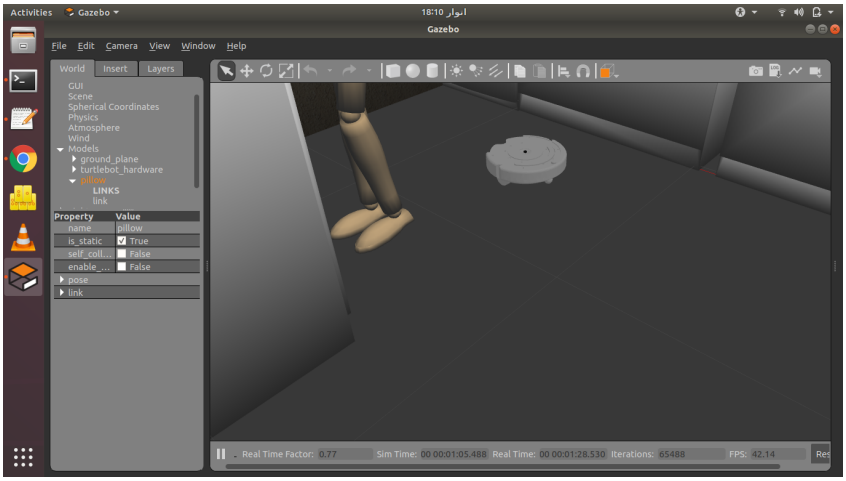


Figure 7.8: Obstacle Avoidance & Moving Towards Goal

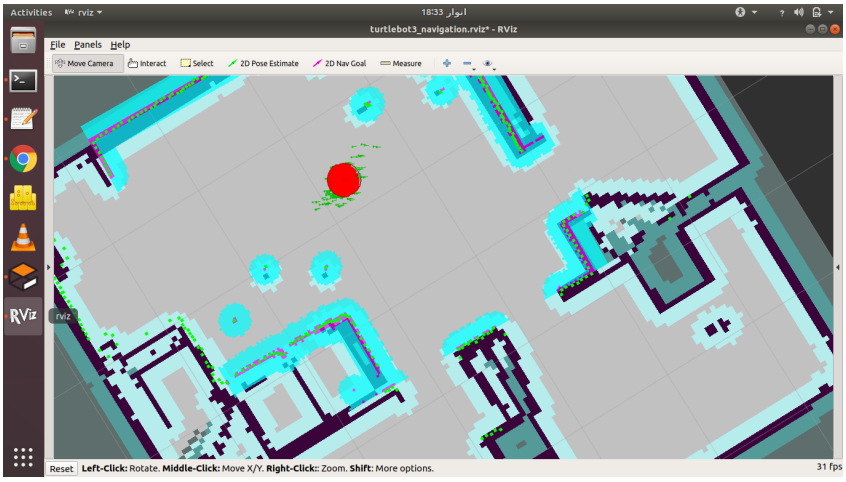


Figure 7.9: Goal Achievement in Autonomous Navigation

Chapter 8

Motion Planning

"The definition of insanity is doing the same thing over and over again and expecting different results."

Albert Einstein

8.1 Cleaning Motion Planning Techniques

If we talk about the autonomous robot, path planning objective is to find the path which is collision free from the start state to to goal state. When it comes to make the path planning in terms of robotic cleaners, it is very complex. As it includes the coverage of whole environment with the application of cleaning the surroundings.

We can design various motion planning algorithms to plan the path with respect to sophistication needed. Some of the robots sophistication demands expensive units of sensors and need heavy computation. While some companies have worked on simple algorithms which includes the bumper sensors and IR sensors for obstacle avoidance and these robots move randomly to cover the path which is very time taking and wastage of time. Below in this chapter we will discuss the efficient motion planning algorithm.

- **Random Coverage** As the name describes, the robot moves randomly to clean the environment until it bumps the wall then it return back at some angle and move again. This algorithm repeats this bumping to obstacles and change in orientation. These robots are cheap and takes too much time to cover the path. It looks simple but complete coverage is impossible. There is too much possibility to repeat the

already traversed regions. These are suitable for domestic cleaning where time has not any periority [49].

- **Systematic or pattern coverage** This algorithm involves the pattern path which is stored in robot memory. It will be simple but not computationally heavy. The systematic coverage of environment is used in this project [49].

Below are the advantages of these type of pattern or systematic motion planning algorithm

- The paths are generated for the coverage of complete area
- It is not necessary to have the complete knowledge of obstacles. SLAM will avoid the obstacles automatically.
- The pattern of following the path made is very efficient to clean the dust from all the areas [50, 51].
- It does not need to save too much data bags which shows that computationally it is not very high price. There are several motion patterns to clean the surroundings e.g. wall follow, zig-zag, spirals etc. In this project we will follow the zig-zag pattern [52].

8.1.1 Zig-Zag Motion Algorithm

In this type of motion robot moves in the straight line then when it reaches the first goal next goal is given at small distance and then it takes turn to get the motion parallel to the previous traversed straight line. The motion is very similar to "L" shape. If we see from any corner it looks simple L and inverted L motion from other end to cover the whole path. Figure 8.1 shows the path generated by a zigzag motion algorithm

8.2 Path Plan Method

An efficient algorithm to clean the region in a small amount of time by making zig-zag path is very feasible. Below shown is the algorithm **??** which is used to make an efficient path.

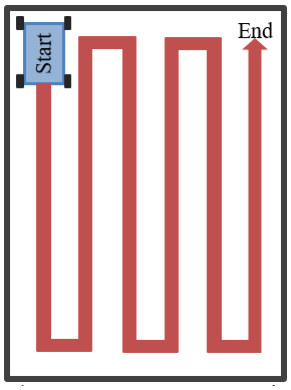


Figure 8.1: Zig-Zag Motion Path

Algorithm 3 Efficient Motion Planning Algorithm Part I

Input: $map(m), P_1, P_2, P_3, P_4, P_5$

Output: path

procedure PLANPATH(P)

$map \leftarrow m$

$polygon \leftarrow P_1, P_2, P_3, P_4, P_5$

$samples \leftarrow points[polygon]$

Function `SelectingEdge(`*polygon*`):`

$CornerDistance \leftarrow \delta_1, \delta_2, \delta_3, \delta_4$

$robot \leftarrow (x_i, y_i)$ ▷ To find the shortest edge

$ExteriorPoints \leftarrow (x_s, y_s)$

$Points \leftarrow zip(x_s, y_s)$

for $i \leftarrow 1$ *to* $range(Points)$ **do**

$pair = (x[s+i], y[s+i])$

$distance = \sqrt{(pair[x][i+1] - pair[x][i])^2 + (pair[y][i+1] - pair[y][i])^2}$

if $\delta < distance$ **then**

$\delta = distance$

$robot = pair(x, y)$

end

end

end Function

return $transformation \leftarrow robot[pair]$

Algorithm 3 Efficient Motion Planning Algorithm Part II

procedure PLANPATH(P)

$threshold \leftarrow th$

$pixel \leftarrow p$

Function `vericalNarrowSpace`(m, th)**:**

```
for all p (x, y) in m do
    if the p (x,y) is free spaced p m(x, y) = 0 then
        v_l ← 0, v_r0
        while pixels (x − v_l, y),(x + v_r, y) are the free spaced pixels do
            v_l ← v_l + 1, v_r ← v_r + 1
        end
        if 0.5 ×th < v_l + v_r < th then
            narrow p (x_ns, y_ns)withx_ns ← x, y_ns ← y
            m(x_ns, y_ns) ← 200
        end
    end
end
```

Function `horizontalNarrowSpace`(m, th)**:**

```
for all p (x,y) in m do
    if the p (x,y) is free spaced p m(x, y) = 0 then
        h_x ← 0, h_y0
        while pixels (x, y − h_y),(x, y_hy) are the free spaced pixels do
            h_x ← h_x + 1, h_y ← h_y + 1
        end
        if 0.5 ×th < h_x + h_y < th then
            narrow p (x_ns, y_ns)withx_ns ← x, y_ns ← y
            m(x_ns, y_ns) ← 200
        end
    end
end
end Function
return narrow p
end Function
```

Algorithm 3 Efficient Motion Planning Algorithm Part III

procedure PLANPATH(P)
Function `Goal(`P_G`):`
    $goalPoint \leftarrow P_G$
    $startPoint \leftarrow P_S$
    $n \leftarrow points$
    for $j \leftarrow 1$ *to* $points$ **do**
        $goalpoint[i] = (x_s + i, y_s + i)$ ▷ Refreshing the goal state
        $shortPath \leftarrow 3$
        $goalpoint[i] = (x_s + [i + shorPath], y_s + [i + shortPath])$
    end
    return $goalpoint \leftarrow P_G$
    $move$base $\leftarrow P_G$
end Function

8.3 Working of Algorithm

- A region is selected which is to be cleaned by four points square.
- Multiple random points are formed, where P(s) is starting point which is determined by distance formula. .
- The Robot finds the shortest distance from its origin to the respective four points. The node with shortest distance is selected.
- Robot then goes to the nearest point of the selected node from where the next point is selected which is nearest point of second node.

In Part II of Algorithm, there is determination of narrow space pixels. The intermediate way-points finding has been started by global planner. The path is determined by path planning algorithm by finding current way-point nearest pixel. The algorithm ensures the safety distance from obstacle while working.

In part III of Algorithm, goal point is assigned to move base to cover the distance from starting position to next way-point. As shown in Figure 8.3 a) The shortest path towards the goal. b) the partial path coverage of the environment. c) skipping the narrow space where the robot cannot go. d) complete covered path.

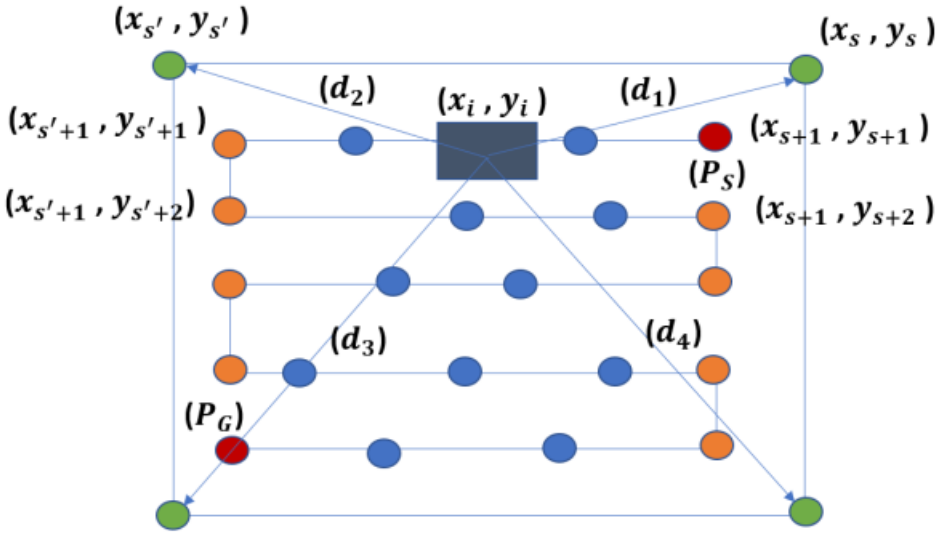


Figure 8.2: Zig-Zag Path Planning

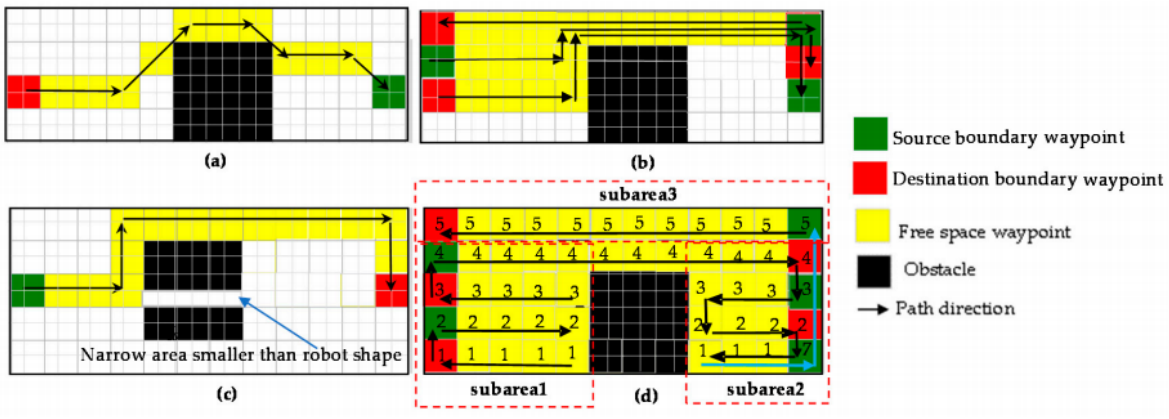


Figure 8.3: Path Coverage Steps

8.4 Working Flow-Chart

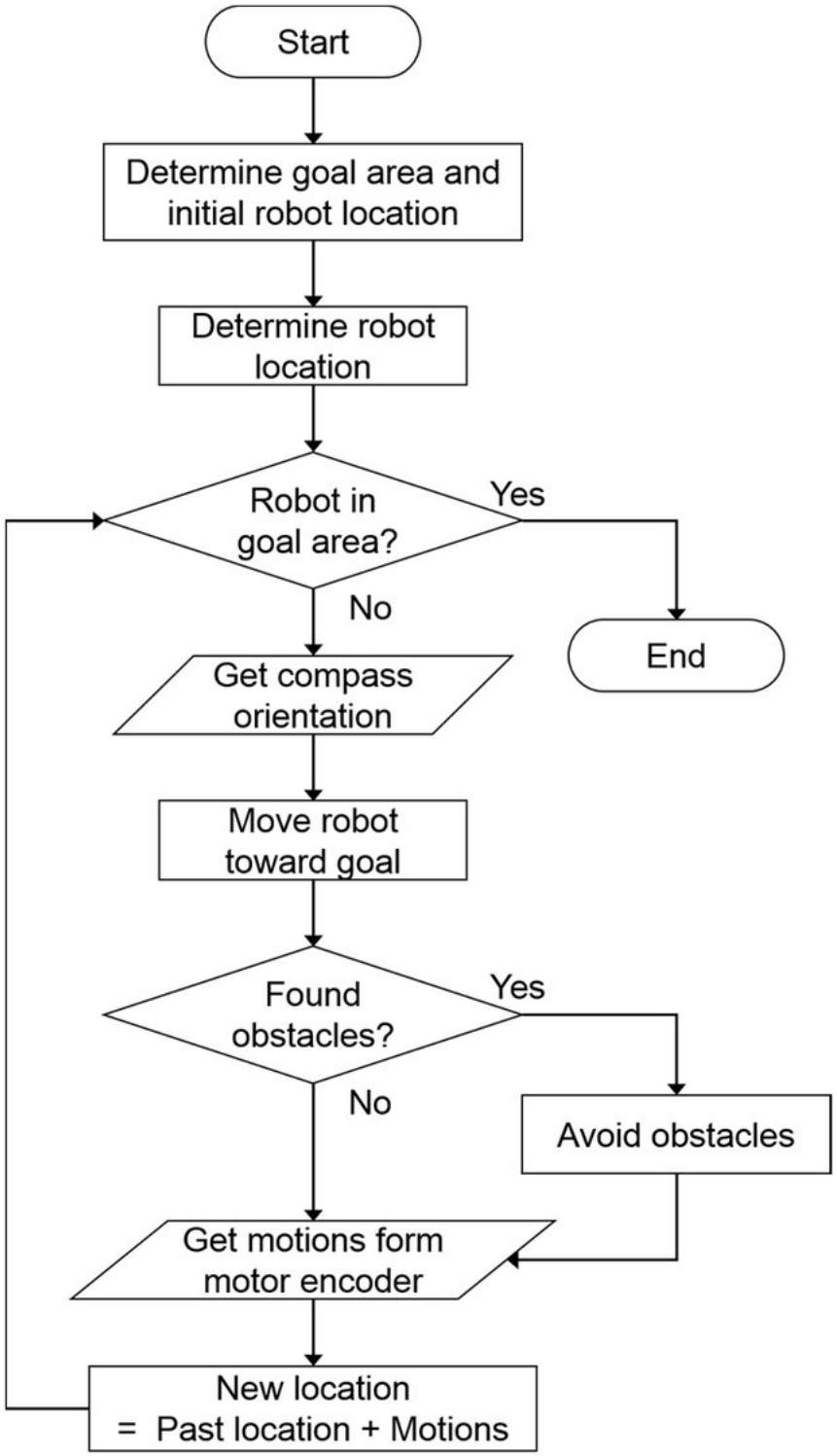


Figure 8.4: Cleaning Robot Autonomous Navigation Flowchart

8.5 Final Working Simulation

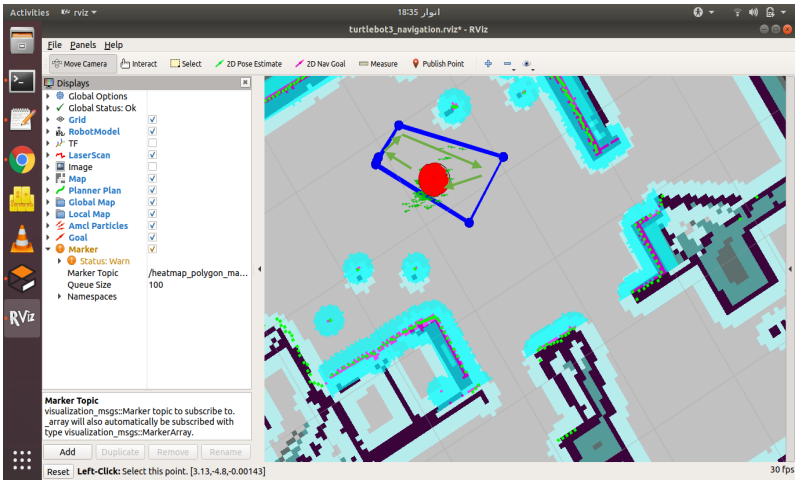


Figure 8.5: Publishing the Polygon

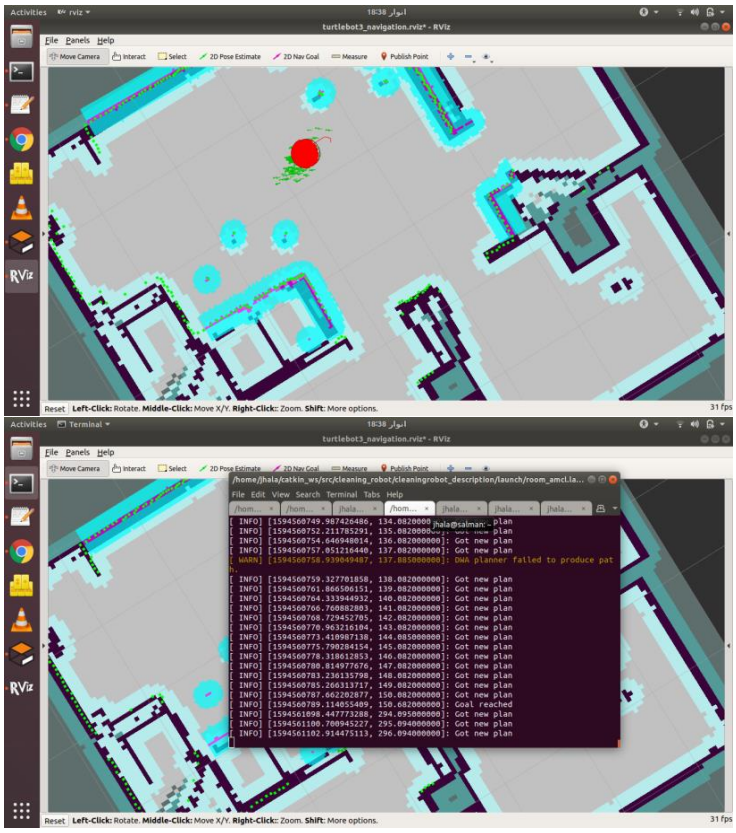


Figure 8.6: Planning and Working of Robot

8.6 Social Impacts & Recommendations

"Cleanliness is half of faith." - Muhammad PBUH

"I only feel angry when I see waste. When I see people throwing away things we could use." - Mother Teresa

Waste materials and dust are the basic problem in Pakistan which can not be neglected. Many people got into different diseases during inhaling and have respiratory problems. They have suffocation problems. So, cleaning the environment is the basic part of every society. Their are a number of group of people who are disabled, having allergy with dust and many other people having no time to waste in cleaning in Pakistan. They want to contribute their efforts to other important work. So, this application is very beneficial for human being. The survey taken by [53] gathers the information about the benefits of systematic cleaning robots. Following questions are raised during this survey

1. How Cleaning benefits the lifestyle?
2. How much time people use these product?
3. How much efficient cleaning robots are?
4. What is the feedback of different persons?
5. Importance of this systematic cleaning of environment?

So, let us start answering the questions one by one. There is infection free lifestyle, and increase self confidence when a person works in clean environment in universities, homes, offices etc. and give high productivity. So, land pollution is totally not acceptable to any one. In survey there are inputs taken from about 386 participants. 88% of people answered yes when the question of saving the time is raised. And 69% of participants have said that by using cleaning robots, their environment is cleaned which results into healthier home environment. The importance of features was achieved by the rating given by respondents between 1 to 6. Below given Figure 8.7 shows the importance of different features.

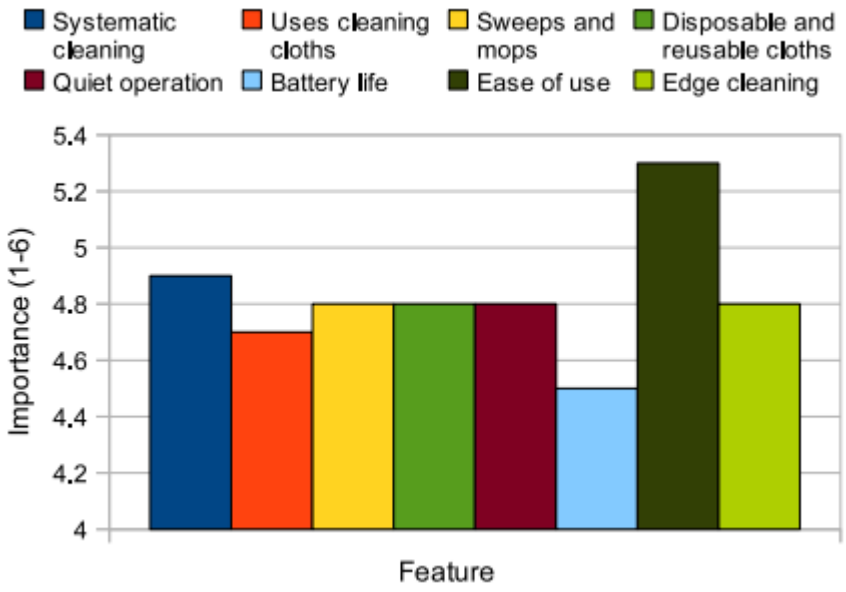


Figure 8.7: Importance of features of cleaning robots [53]

The average rating is shown in the chart in which most important is this these robots are easy to use. The 2nd important thing is its systematic and planned cleaning. And other three i.e. sweeps & mops, edge cleaning and quite operations are very close and on 3rd number of importance. The other features are battery life and disposables which are of less importance. So, cleaning robots are mostly used to save time and effort of a person.

In this survey 65% of people said that they like the new technology and these type of cleaning robots have impacted their life. Many of the people can save their time and effort for some other important work.

It is recommended that their need to optimize the motion planning algorithms to clean the environment. This will save the time, battery and most important effort of a person. The robot must not harm humans or any other living being.

Chapter 9

Conclusions

"Words, without power, is mere philosophy."

Allama M. Iqbal

Cleaning Robots are gaining popularity day by day. In this busy world, every person want to save his time. So, cleaning is the task which is very time taking and make a person very tired. In this thesis we have given the proposed the solution to this problem. Many robots exists in the market and they also provide many features like waxing, sterilization etc. However, they move randomly and do not effectively create map and plan the paths of cleaning. In this project, we have used the ROS tools and frameworks to create an efficient cleaning path planning and user friendly system to increase the capabilities of systems. First of all we have created the robot model with proper differential drive in ROS.

The autonomous mapping technique is used to create map autonomously using frontier based algorithm. The autonomous navigation algorithm is implemented using A* Search based algorithm. The map is tagged like if the command kitchen is given to the tagging algorithm the robot will reach the kitchen autonomously. The efficient Cleaning Robot Motion planning algorithm is created to move the robot in zig-zag path.

There are two mode of controlling the robot either by manual or autonomous. In autonomous mode the user will give the four points on the map which makes the polygon and robot will clean that region with an efficient zig zag algorithm.

9.1 Future Work

In future, we will work on the optimization of algorithm and to increase the features of algorithm like if user give command of kitchen cleaning, the robot will clean the whole kitchen. We can increase our research on this robot to increase its features to spying and security intelligence. its application can be increased to rescue purposes. And of-course for navigation tasks for picking and moving the object from one place to others. These types of robots are very useful in any field of mobile and social robots.

References

[1] A. Z. Christopher and A. Jones, 'Autonomous surface cleaning robot for dry cleaning', *U.S. Patent 8782848 B2*, 2014 (cit. on pp. 1, 2).

[2] R. E. R. N.Terrones, 'Development of a control platform for the mobile robot roomba using ros and a kinect sensor', *Robotics Symposium and Competition, 2013 Latin American*, vol. 55, no. 60, pp. 21–27, 2013 (cit. on p. 1).

[3] R. Kurazume and S. Hirose, 'Development of a cleaning robot system with cooperative positioning system', *Autonomous Robots*, vol. 9, no. 3, pp. 237–246, 2000 (cit. on p. 1).

[4] iRobot. (2019). Irobot fast facts, [Online]. Available: `http://www.irobot.com/EngineeringAwesome/images/iAdapt%5C%20Fast%5C%20Facts.pdf` (cit. on p. 1).

[5] K. M. H. Abdullah-Al-Nahid and K. J. Reza, 'Path planning algorithm development for autonomous vacuum cleaner robots', *3rd INTERNATIONAL CONFERENCE ON INFORMATICS, ELECTRONICS VISION 2014*, 2014 (cit. on p. 1).

[6] L. B. Gangadharaswamy, 'Design of autonomous cleaning robot', *Master of Science Thesis*, 2019 (cit. on p. 2).

[7] H. S. Land. (2011). Hands-on: Neato xv-11 robotic vacuum review, [Online]. Available: `http://www.homeserverland.com/2011/01/hands-on-neato-xv-11-robotic-vacuum-review/` (cit. on p. 2).

[8] BotJunkie. (2010). Neato robotics xv-11, [Online]. Available: `http://www.botjunkie.com/2010/05/18/botjunkie-reviewneato-robotics-xv-11/` (cit. on p. 2).

[9] *Robot operating system (ros)*. [Online]. Available: `https://www.ros.org` (cit. on p. 2).

[10] U. Khalid and M. F. Baloch, 'Smart floor cleaning robot', *Faculty of Electronic Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Pakistan*, vol. 2, no. 8, 2016 (cit. on pp. 11–13).

[11] S. M. K. A. Manjusha, 'Design and implementation of smart floor cleaning robot using android app', *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, 2019 (cit. on p. 11).

[12] T. EDWARDS, *A comparison of path planning algorithms for robotic vacuum cleaners*, 2015 (cit. on p. 11).

[13] X. L. X. Luo and X. Zhong, 'Research on simultaneous localization and mapping of indoor mobile robot', *School of Mechatronic Engineering, Beijing Institute of Technology, Beijing 100081, China*, vol. 3, 2003 (cit. on pp. 12, 14, 15, 18).

[14] yonomi, *Irobot roomba s9+*. [Online]. Available: `https://www.yonomi.co/supported-devices/irobot-roomba-s9-plus` (cit. on pp. 12, 21).

[15] E. Guizzo. (2016). Robots braava jet mopping robot is small, smart, and not round, [Online]. Available: `https://spectrum.ieee.org/automaton/robotics/home-robots/irobot-braava-jet-mopping-robot` (cit. on p. 12).

[16] P. Sayer. (2017). Neato robotics adds more smarts to its vacuum cleaners, [Online]. Available: `https://www.techhive.com/article/3221805/neato-robotics-adds-more-smarts-to-its-vacuum-cleaners.html` (cit. on p. 13).

[17] Y.-Y. CHENG and D.-X. SHI, 'Ros based remote robot task monitoring and control system', *National Laboratory for Parallel and Distributed Processing*, 2017 (cit. on p. 15).

[18] 'Mobile robot localization in a manufacturing environment', *3rd International Conference on Manufacturing Engineering*, vol. 5, 2008 (cit. on p. 15).

[19] A. V. R. M. F. M. Angermann and J. Mueller, 'Autonomous robotic slam-based indoor navigation for high resolution sampling with complete coverage', *Michael Lichtenstern Institute of Communications and Navigation*, 2016 (cit. on p. 16).

[20] D. W. M. K. J. W. D. N. Onderr, 'Autonomous navigation planning with ros', *MICHIGAN TECHNOLOGICAL UNIVERSITY*, vol. 5, 2017 (cit. on p. 18).

[21] frederico C.VIEIRA and Adelardo, 'Position and orientation control of a two-wheel differential drive nonholonomic mobile robot', *federal University of Rio Grande do Norte, Brazil*, (cit. on p. 20).

[22] C. Myint and N. N. Win, 'Position and velocity control for two-wheel differential drive mobile robot', *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 5, no. 9, 2016 (cit. on p. 20).

[23] T. Hellström, 'Kinematics equations for differential drive and articulated steering', *Department of Computing Science Umea University*, 2011 (cit. on pp. 20, 24).

[24] Dudek and Jenkin, *Cs w4733 notes - differential drive robots*, 2016 (cit. on p. 21).

[25] G. Walck, *Introduction to robot modeling in ros*, 2015 (cit. on p. 29).

[26] R. wiki. (2019). Urdf/xml/joint, [Online]. Available: `https://namesdir.com/mirrors/wiki.ros.org/urdf(2f)XML(2f)joint.html` (cit. on p. 29).

[27] Y. Pyo and H. Cho, *ROS Robot Programming*. Seoul, Republic of Korea: Robotis Co., 2017 (cit. on pp. 29, 35, 36).

[28] N. Castaman, *3d robot modeling and simulation in ros*, 2018 (cit. on p. 31).

[29] Robotis. (2020). Lds-01, [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/appendix_lds_01/ (cit. on p. 37).

[30] Elprocus. (2020). Imu sensor working and its applications, [Online]. Available: https://www.elprocus.com/imu-sensor-working-applications/ (cit. on p. 37).

[31] H. ullah sharif hossain and bulbul islam, 'Room mapping and automatic vacuum cleaner robot', *ORGANIZATION OF ISLAMIC CONFERENCE*, 2019 (cit. on p. 45).

[32] C. M. KOEHR, 'Evaluation of ros slam gmapping', *The University of Alabama*, 2018 (cit. on p. 47).

[33] R.Korb and A.Schottl, 'Exploring unstructured environment with frontier trees', *Journal of Intelligent & Robotic Systems*, 2017 (cit. on p. 48).

[34] C. Gomez and A. Hernandez, 'Topological frontier based exploration and map building using semantic information', *Roboticslab Research Group*, 2019 (cit. on pp. 48, 51).

[35] A. Kathpal and P. Inani, 'Frontier based exploration for autonomous robot', *Maryland Robotics Center*, 2019 (cit. on pp. 48, 49).

[36] A. A. M. Amasyalo and S. Yavuz, 'Implementation of frontier-based exploration algorithm for an autonomous robot', *IEEE*, 2015 (cit. on p. 51).

[37] D. D. S. T. M. Montemerlo and J. Diebel, 'Path planning for autonomous vehicles in unknown semi-structured environments', *International Journal of Robotics Research*, vol. 29, pp. 485–501, 2010 (cit. on pp. 53, 54).

[38] K. A. S. Nilwong, 'Robot navigation in outdoor environments using odometry and convolutional neural network', *IEEJ International Workshop on Sensing, Actuation, Motion Control, and Optimization (SAMCON)*, vol. 1, 2019 (cit. on p. 53).

[39] S. G. W.Burgard, 'A fully autonomous indoor quadrotor', *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012 (cit. on p. 53).

[40] E.-E. E. T. B. K.Konolige, 'The office marathon: Robust navigation in an indoor office environment', *2010 IEEE International Conference on Robotics and Automation (ICRA12)*, pp. 300–307, 2010 (cit. on p. 53).

[41] S. C.Hernandez X.Sun and P. Meseguer, 'Tree adaptive a*', *International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS11)*, pp. 123–130, 2011 (cit. on p. 54).

[42] N. S.Shen and V.Kumarr, 'Stochastic differential equation-based exploration algorithm for autonomous indoor 3d exploration with a micro-aerial vehicle', *International Journal of Robotics Research*, vol. 31, pp. 1431–1444, 2012 (cit. on p. 54).

[43] N. Roy, 'Autonomous navigation in unknown environments using machine learning', *Massachusetts Institute of Technology*, vol. 1, 2017 (cit. on p. 54).

[44] wekipedia. (2015). A* search algorithm, [Online]. Available: `https://en.wikipedia.org/wiki/A*%5C_search%5C_algorithm` (cit. on p. 55).

[45] B. Sturm. (2020). Demonstration of astar (a*), matlab central file exchange, [Online]. Available: `https://www.mathworks.com/matlabcentral/fileexchange/26723-demonstration-of-astar-a` (cit. on p. 56).

[46] L. Galtarossa, 'Obstacle avoidance algorithms for autonomous navigation system in unstructured indoor areas', *POLITECNICO DI TORINO*, vol. 1, 2018 (cit. on pp. 57, 58).

[47] A. S. de Oliveira and V. A. Brenner, 'Ros navigation: Concepts and tutorial', *Studies in Computational Intelligence*, 2018 (cit. on p. 57).

[48] E. Z. Gomez, 'Map-building and planning for autonomous navigation of a mobile robot', *Center for Research and Advanced Studies of the National Polytechnic Institute*, 2015 (cit. on p. 58).

[49] V. P. Anh Vu Le and V. Sivanantham, 'Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor', *Optoelectronics Research Group*, vol. 1, 2018 (cit. on p. 63).

[50] A.-A.-N. Kazi Mahmud Hasan, 'Planning algorithm development for autonomous vacuum cleaner robots', *3rd INTERNATIONAL CONFERENCE ON INFORMATICS, ELECTRONICS VISION 2014*, 2018 (cit. on p. 63).

[51] Y. Y. Xiangrong Xu and S. Pan, 'Motion planning for mobile robots', *IntechOpen*, 2018 (cit. on p. 63).

[52] R. Cesar, 'Planning in the context of active localization and multi-hypothesis tracking', *Charles University*, 2018 (cit. on p. 63).

[53] J.-S. G. K. C. M. E. P. Pirjanian, 'The social impact of a systematic floor cleaner', *3rd INTERNATIONAL CONFERENCE 2012*, 2012 (cit. on pp. 70, 71).

Appendices

.1 Gazebo Plugins

- **LiDAR Plugin**

```
<gazebo reference="hokuyo_link">
  <material>Gazebo/FlatBlack</material>
  <sensor type="ray" name="lds_lfcd_sensor">
    <pose>0 0 0 0 0 0</pose>
    <update_rate>5</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>360</samples>
          <resolution>1</resolution>
          <min_angle>0.0</min_angle>
          <max_angle>6.28319</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.120</min>
        <max>3.5</max>
        <resolution>0.015</resolution>
      </range>
      <noise>
        <type>gaussian</type>
        <mean>0.0</mean>
        <stddev>0.01</stddev>
      </noise>
    </ray>
    <plugin name="gazebo_ros_lds_lfcd_controller"
filename="libgazebo_ros_laser.so">
      <topicName>scan</topicName>
      <frameName>hokuyo_link</frameName>
    </plugin>
  </sensor>
</gazebo>
```

- **Differential Drive Plugin**

```
    <plugin name="differential_drive_controller"
filename="libgazebo_ros_diff_drive.so">
      <commandTopic>cmd_vel</commandTopic>
      <odometryTopic>odom</odometryTopic>
      <odometryFrame>odom</odometryFrame>
      <odometrySource>world</odometrySource>
      <publishOdomTF>true</publishOdomTF>
      <robotBaseFrame>base_footprint</robotBaseFrame>
      <publishWheelTF>false</publishWheelTF>
      <publishTf>true</publishTf>
      <publishWheelJointState>true</publishWheelJointState>
      <legacyMode>false</legacyMode>
      <updateRate>30</updateRate>
      <leftJoint>left_wheel_joint</leftJoint>
      <rightJoint>right_wheel_joint</rightJoint>
      <wheelSeparation>0.26</wheelSeparation>
      <wheelDiameter>0.066</wheelDiameter>
      <wheelAcceleration>1</wheelAcceleration>
      <wheelTorque>10</wheelTorque>
      <rosDebugLevel>na</rosDebugLevel>
    </plugin>
  </gazebo>
```

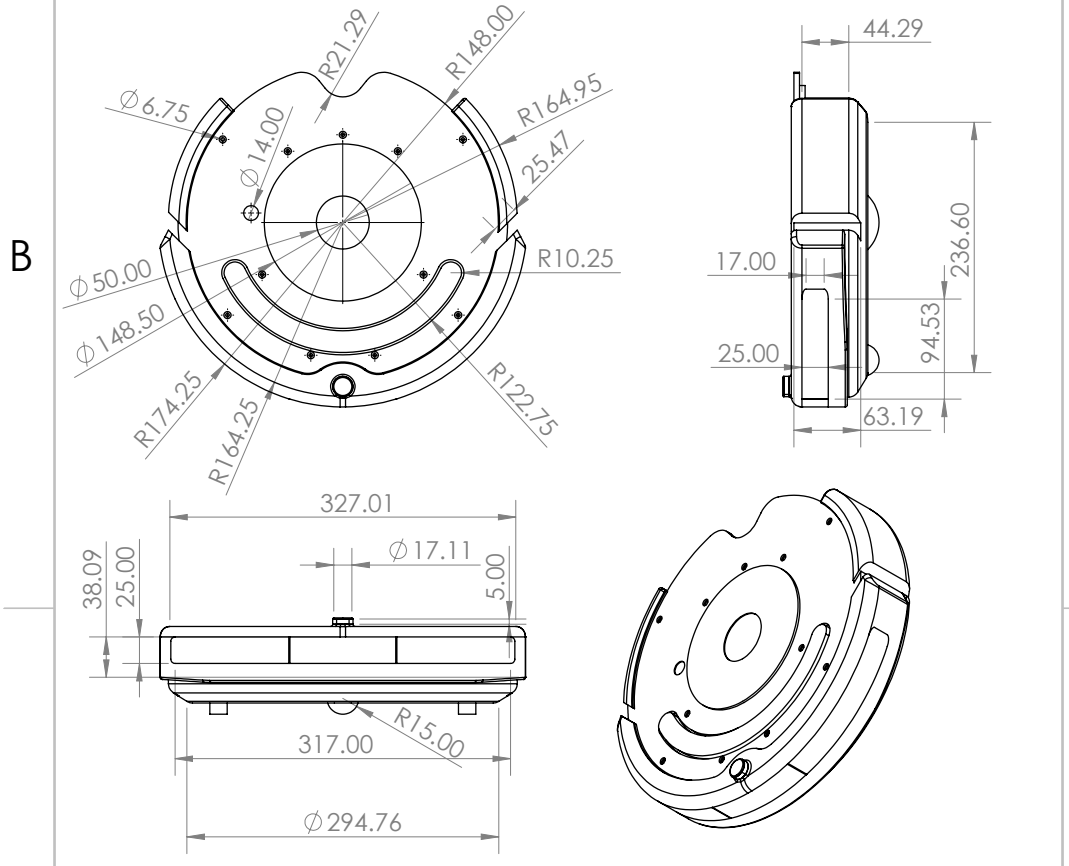
- **IMU Plugin**

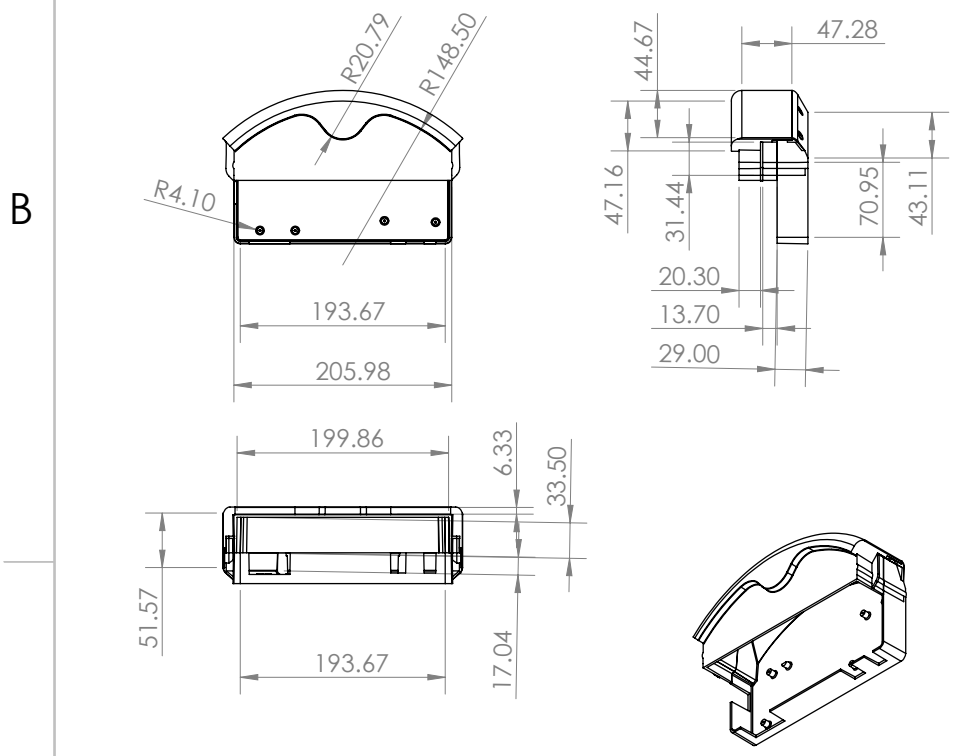
```
<plugin name="imu_plugin" filename="libgazebo_ros_imu.so">
  <alwaysOn>true</alwaysOn>
  <bodyName>imu_link</bodyName>
  <frameName>imu_link</frameName>
  <topicName>imu</topicName>
  <serviceName>imu_service</serviceName>
  <gaussianNoise>0.0</gaussianNoise>
  <updateRate>200</updateRate>
  <imu>
    <noise>
      <type>gaussian</type>
      <rate>
        <mean>0.0</mean>
        <stddev>2e-4</stddev>
        <bias_mean>0.0000075</bias_mean>
        <bias_stddev>0.0000008</bias_stddev>
      </rate>
      <accel>
        <mean>0.0</mean>
        <stddev>1.7e-2</stddev>
        <bias_mean>0.1</bias_mean>
        <bias_stddev>0.001</bias_stddev>
      </accel>
    </noise>
  </imu>
</plugin>
```

- **Joint State Publisher Plugin**

```
<plugin name="joint_state_publisher"
filename="libgazebo_ros_joint_state_publisher.so">
<jointName>hokuyo_joint, imu_joint,
front_castor_joint, rear_castor_joint,
base_rightfront_cliff_sensor_joint,
base_leftfront_cliff_sensor_joint,
base_right_cliff_sensor_joint,
base_left_cliff_sensor_joint,
base_wall_sensor_joint,
base_footprint_joint</jointName>
</plugin>
```

.2 Engineering Drawing





		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±		NAME	DATE
			DRAWN		
			CHECKED		
			ENG APPR.		
			MFG APPR.		
			Q.A.		
		MATERIAL	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

PROPRIETARY AND CONFIDENTIAL

THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <COMPANY NAME >. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <COMPANY NAME> IS PROHIBITED.

SIZE	DWG. NO.	REV.
A	A2binbottomwithoutcaster	
SCALE:1:5	WEIGHT:	SHEET 1 OF 1

.3 Code

- **Exploration Function**

```
void Explore::makePlan()
{
  // find frontiers
  auto pose = costmap_client_.getRobotPose();
  // get frontiers sorted according to cost
  auto frontiers = search_.searchFrom(pose.position);
  ROS_DEBUG("found %lu frontiers", frontiers.size());
  for (size_t i = 0; i < frontiers.size(); ++i) {
    ROS_DEBUG("frontier %zd cost: %f", i, frontiers[i].cost);
  }

  if (frontiers.empty()) {
    stop();
    return;
  }

  // publish frontiers as visualization markers
  if (visualize_) {
    visualizeFrontiers(frontiers);
  }

  // find non blacklisted frontier
  auto frontier =
      std::find_if_not(frontiers.begin(), frontiers.end(),
                       [this](const frontier_exploration::Frontier& f) {
                         return goalOnBlacklist(f.centroid);
                       });
  if (frontier == frontiers.end()) {
    stop();
    return;
  }
```

- **Path Planning Functions**

```
def generate_intersections(poly, width):
    "Subdivide a filed into coverage lines."
    starting_breakdown = poly.bounds[0:2]
    line = LineString([starting_breakdown, (starting_breakdown[0],
                                            starting_breakdown[1] +
                                            poly.bounds[3] - poly.bounds[1])])
    try:
        bounded_line = poly.intersection(line)
    except TopologicalError as e:
        error("Problem looking for intersection.", exc_info=1)
        return
    lines = [bounded_line]
    iterations = int(math.ceil((poly.bounds[2] - poly.bounds[0]) / width)) + 1
    for x in range(1, iterations):
        bounded_line = line.parallel_offset(x * width, 'right')
        if poly.intersects(bounded_line):
            try:
                bounded_line = poly.intersection(bounded_line)
            except TopologicalError as e:
                error("Problem looking for intersection.", exc_info=1)
                continue
            lines.append(bounded_line)
    return lines
```

```
def order_points(lines, initial_origin):
    "Return a list of points in a given coverage path order"
    origin = initial_origin
    results = []
    while True:
        if not len(lines):
            break
        lines = sort_to(origin, lines)
        f = lines.pop(0)
        if type(f) == GeometryCollection:
            continue
        if type(f) == MultiLineString:
            for ln in f:
                lines.append(ln)
            continue
        if type(f) == Point or type(f) == MultiPoint:
            continue
        xs, ys = f.xy
        ps = zip(xs, ys)
        (start, end) = get_furthest(ps, origin)
        results.append(origin)
        # results.append(start)
        results.append(start)
        # results.append(end)
        origin = end
    return results
```

.4 Gantt Chart

WORK GANTT CHART

Development of efficient motion planning algorithm	Group 2	8/01/2019	7/10/2020
PROJECT NAME	OWNER	START DATE	END DATE

Description
This chart was initiated to maintain the project schedule

Task Name	Days	AUG	SEP	OCT	NOV	DEC	JAN	FEB	MAR	APR	MAY	JUN	JUL
Data Gathering	28												
Literature Review and ROS courses	35												
Project Proposal	25												
3D CAD Modeling	34												
URDF/GAZEBO simulations	49												
Autonomous Mapping/SLAM	30												
Algorithm Development Phase 1	35												
Algorithm Development Phase 2	28												
Review and Testing	31												
Thesis & Final year Presentation	30												

Table 1: Implementation Plan

.5 Team Charter

Team Purpose

This team has performed very well to achieve the research task & final product. This product will be beneficial in future.

Duration & Time Commitment

This team has worked 8 months to achieve this research task. The weekly effort was 60% of total working hours

Members

- Muhammad Faizan Ali Khan
- Abdul Anan
- Hamza Habib
- Salman Khan

Desired End Results

This product will minimize the laborious effort for cleaning purposes. The team has completed the project and has almost got result nearly equals to 80%. That are much closer to required results.

Reporting Plan

Every project need a routine work, thats why this team worked on daily bases and has reported the concerning Supervisor every week & has checked/submitted their performances

.6 Closeout Memos

PROJECT CLOSEOUT MEMO

Dear Project Coordinator:

Mr. Muhammad Faizan Ali Khan has completed the project **Development of an Efficient Motion Planning Algorithm For Cleaning Robot.**

I certify the following:

- the project is complete and all state reimbursement claims are settled;
- the project is completely in working;
- student followed all Public law in the manufacturing of the project;
- as per university requirements, student will retain all project financial records for a period of no less than 01 year.

Sincerely,

Project Supervisor
Dr. Ahmed Nouman

PROJECT CLOSEOUT MEMO

Dear Project Coordinator:

Mr. Abdul Anan has completed the project **Development of an Efficient Motion Planning Algorithm For Cleaning Robot.**

I certify the following:

- the project is complete and all state reimbursement claims are settled;
- the project is completely in working;
- student followed all Public law in the manufacturing of the project;
- as per university requirements, student will retain all project financial records for a period of no less than 01 year.

Sincerely,

_____

Project Supervisor
Dr. Ahmed Nouman

PROJECT CLOSEOUT MEMO

Dear Project Coordinator:

Mr. Hamza Habib has completed the project **Development of an Efficient Motion Planning Algorithm For Cleaning Robot.**

I certify the following:

- the project is complete and all state reimbursement claims are settled;
- the project is completely in working;
- student followed all Public law in the manufacturing of the project;
- as per university requirements, student will retain all project financial records for a period of no less than 01 year.

Sincerely,

_____

Project Supervisor
Dr. Ahmed Nouman

PROJECT CLOSEOUT MEMO

Dear Project Coordinator:

Mr. Salman Khan has completed the project **Development of an Efficient Motion Planning Algorithm For Cleaning Robot.**

I certify the following:

- the project is complete and all state reimbursement claims are settled;
- the project is completely in working;
- student followed all Public law in the manufacturing of the project;
- as per university requirements, student will retain all project financial records for a period of no less than 01 year.

Sincerely,

_____

Project Supervisor
Dr. Ahmed Nouman