

# Turtlebot Maze Solving With ROS

N Harshavardhan Reddy, G Vamshi Krishna, Shaik Shahid Ali

School of Electronic & Electrical Engineering

VIT University, Vellore, India

harshavardhan.n2019@vitstuent.ac.in, gollavamshi.krishna2019@vitstudent.ac.in, shaikshahid.ali2019@vitstudent.ac.in

**Abstract-** In this project, we are building a maze solver using Turtlebot. It is predicted that usage of automated robotic systems will increase tremendously in both industrial and domestic applications such as path planning bot, robot-based room cleaning system, robot-based waiter, etc. Currently, during the pandemic, it has become important to minimize human-to-human contact to stop transmission of disease hence there is a necessity to use them for pathfinders such as in restaurants. As a result, there is a need for robots to be equipped with this technology. Due to this, we have decided to find an approach for a bot to find to navigate and move the bot from one place to another automatically. For this approach, we have used ROS, modified Turtlebot package, and Gazebo. After compiling and integrating various nodes using ROS we were successfully able to simulate bot which could solve a maze and avoid obstacles in its path.

**Keywords-** ROS, Simulation, Robotics, Turtlebot.

## I. INTRODUCTION

Navigating through unstructured environments is a basic capability of intelligent creatures, and thus is of fundamental interest in any robotics industry. Robots and automated systems are ideally suited to take on repetitive, dirty, and dangerous jobs. These systems have already proven their value in commercial environments.

Therefore, we aim to understand the reason behind the excessive research and usage in the Robotics industry by simulating the autonomous navigation capabilities of the Turtlebot which in turn could have an application of a service bot such as food delivery bot in the future. We make use of the ROS Kinetic and Gazebo environment to simulate the same.

## II. OBJECTIVE

As mentioned earlier, autonomous navigation is a feature that would prove to be useful for most mobile robots, especially for service robots. In the situation that we are in right now, no-contact essential services are required the most. One of the essential services is food delivery and autonomous bots can make that happen. In this project, we've used a maze to test the autonomous navigation skills of a Turtlebot through ROS and Gazebo. Autonomous navigation can only be achieved upon performing vigorous testing of algorithms used.

In our simulation, we test all the algorithms that the bot uses to compute an obstacle free path. In the real world there is always a possibility of errors to occur while computing the path, therefore we've also included a PID program to rectify errors at all times. Overall, we chose this project so that we could understand the concept of

autonomous navigation and also learn ROS and its features.

## III. RESEARCH AND UNDERSTANDING

Initially we had research on whether ROS Turtlebot would be a good choice for simulating a simple autonomous navigation system. We used the information from a few YouTube channels to do the same. Next, we had to research on how ROS works. After spending hours of watching YouTube tutorials, we figured out that the best version of ROS to simulate the Turtlebot is ROS Kinetic and therefore we proceeded with the installation process.

Since dual booting didn't work, we went on with installing Ubuntu 16.04 in VirtualBox and then installed ROS Kinetics along with its dependencies. After the installation we had to learn about the basics of interacting with the command line of Ubuntu. We took help from YouTube tutorials for the same. After we've gained sufficient knowledge of the basics, we proceeded further with our project.

## IV. BASICS OF ROS

After setting up the ROS Kinetic environment in Ubuntu 16.04, we had to learn the basics of ROS to proceed further with the project. We started with learning how to navigate in the ROS file system. A catkin tool is the python package that provides a command line interface for the ROS file system, therefore understanding it was crucial.

All project files are to be placed in the catkin workspace folder. Next, we learnt how to build a ROS package using the installed Catkin tools. Since the vital part of our project is the Turtlebot, we installed its package and dependencies using the methods that we learnt until that point. Next, we

learnt about ROS Nodes and Topics. ROS nodes are basically processes that perform computation. Each node in a ROS system has a unique name and they communicate with each other through ROS Topics, services, actions etc. Next, we learnt about Publishing and Subscribing, which are the processes through which ROS Nodes communicate with each other. In our project, there are various ROS Nodes/Topics involved. The important nodes are odom, laserscan, and Maze solver Node. Each of these nodes parallelly computes and communicates with each other at the same time. To visualize all the nodes and topics used, rqt\_console was used. It helped in providing a rqt\_graph which is essentially a graph that shows all the nodes and topics involved in the system.

## V. ABBREVIATIONS AND ACRONYMS

- SLAM – Simultaneous Localization and Mapping. A laser based scan for creating a 2-D map.
- LTS – Long Term Support. Software support for Ubuntu OS
- ROS – Robot Operating System. An open source programming robot software units.
- PID – Proportional Integral Derivative

### 1. Turtlebot:

The most important part of this project is the Turtlebot. It is a fairly small mobile robot that can programmed to perform various operations. Turtlebots come with a lot of sensors built-in. Since the aim of this project is get the turtlebot to solve a maze on its own, we made use of the 2D/3D distance sensors available on the Turtlebot. The data received from the distance sensor was used for computation in the laserscan ROS Node which was mentioned earlier. Gazebo package is required to simulate the turtlebot's environment. Gazebo is a simulation tool that helps simulate a real time environment for the robot and also perform various tests to check the working of a robot. Since our Turtlebot requires a maze, we've simulated the same in Gazebo.

### 2. Gazebo:

Simulator Gazebo is simulation software collaborated to ROS that provides 3D dynamic simulator. This enables to simulate efficiently and accurately robot's operation in complex outdoor and indoor environments. A typical use of Gazebo is to test robotics algorithms, robot designs and executing testing in realistic scenarios

### 3. Proportional Integral Derivative Controller (PID):

Since there is a possibility of occurrence of errors in the computation, we use a PID controller package to rectify it. A PID controller is a control loop mechanism employed feedback that is widely used in the industries. It constantly calculates the error terms and provides a correction based on the proportional, integral and derivative terms, hence the name PID. ROS provides a PID package which helps performing the same function a physical PID controller.

Tuning the PID control was a tedious task in ROS as the PID package used in ROS Kinetic was a beta release.

## VI. METHODOLOGY

The Turtlebot is programmed to identify obstacles, loops and also walls. According to the algorithm, when the bot is in "Walledetection" state, it searches for walls and moves towards them. If an obstacle is found instead of wall, the bot rotates 90 degrees towards its left and goes back to the "Walledetection" state. After the bot goes towards a wall, it goes into the "Wallfollow" state, where it follows the wall until the next opening.

Since it is possible for the bot to fall into a loop, it also keeps track of all the coordinates while its moving, so that if it comes back to the same coordinates, it will acknowledge the detected loop and search for the next wall to move to. The "Wallfollow" state uses the PID controller methods to perform error free navigation. The PID controller essentially adjusts the direction of the Turtlebot. Now that we've understood the working of the algorithms, we simulated the project in Gazebo and achieved the expected result

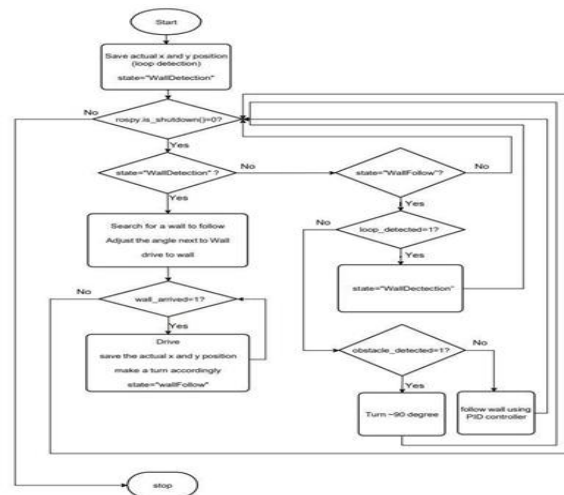


Fig 1. Diagram.

### 1. Gazebo Environment:

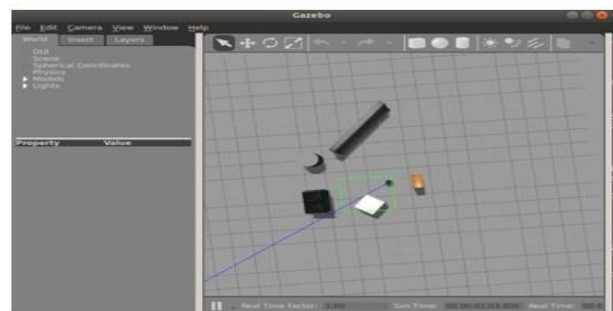


Fig 2. Gazebo Environment.

Here we launched the Turtlebot onto the gazebo environment without adding any maze to the gazebo environment. The Turtlebot emits the blue rays, the laser sensor that we used for the wall detection for autonomous navigation and solving the maze.

## 2. Maze:

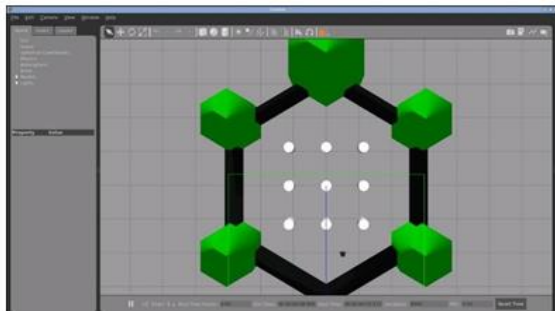


Fig 3. Maze.

## 3. Scanning Roadmap Of Maze:

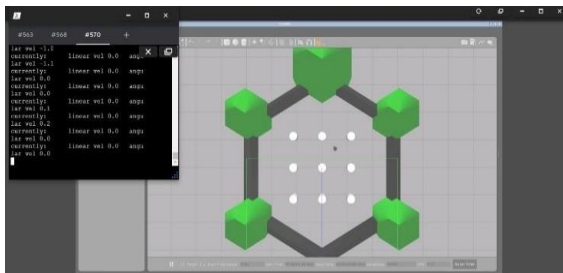


Fig 4. Scanning Roadmap Of Maze:

## 4. Overview of the Roadmap:

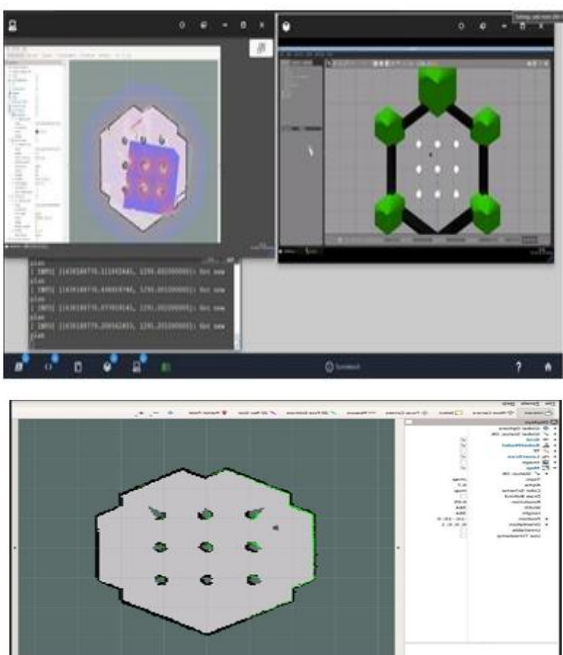


Fig 5. Overview of the Roadmap.

## 5. Turtlebot Choosing Shortest Path:

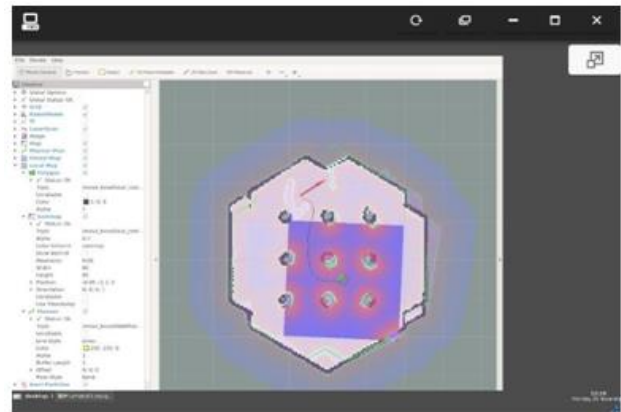


Fig 6. Turtlebot Choosing Shortest Path:

## VII. FUTURE WORKS

The Turtlebot package that we've used in our project is a relatively older version; a newer version like the Turtlebot3 could be used to improve the efficiency of the system. The algorithms used for this system prove to be effective in a relatively less obstacle free area, it needs to be improved in order to be used commercially.

## VIII. LIMITATIONS

In this project, we had simulated the working of Turtlebot maze solving using ROS Kinetic version. So, our project will not be supported on the latest versions of ROS. Although the Turtlebot can navigate good enough using the inbuilt 3D distance sensor, it is still not very accurate and hence cannot be used in the real environment. The Turtlebot's mobility is not the very best, it takes a fairly large amount of time to rotate therefore we had to force pause the code every time it has to rotate during its navigation. Another major drawback of this system is that if any of the nodes doesn't respond, there are no fail-safe methods programmed.

## REFERENCES

- [1] Haber, A., McGill, M., Sammut, C.: jmesim: An open source, multi platform robotics simulator. In: Proceedings of Australasian Conference on Robotics and Automation, New Zealand, 3-5 December. (2012)
- [2] Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software. (2009)
- [3] Milstein, A., McGill, M., Wiley, T., Salleh, R., Sammut, and C.: A method for fast encoder-free mapping in unstructured environments. Journal of Field Robotics 28 (2011) 817–831
- [4] Zaman, S., Slany, W., Steinbauer, G.: Ros-based mapping, localization and autonomous navigation

- using a pioneer 3-dx robot and their relevant issues. In: Electronics, Communications and Photonics Conference (SIEPC), 2011 Saudi International. (2011)
- [5] Grisetti, G., Stachniss, C., Burgard, W.: Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on* 23 (2007)
- [6] Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., Konolige, K.: The office marathon: Robust navigation in an indoor office environment. In: *IEEE International Conference on Robotics and Automation (ICRA)*. (2010) 300–307.
- [7] Khusainov, R., Shimchik, I., Afanasyev, I., Magid, E.: Toward a human-like locomotion: Modelling dynamically stable locomotion of an anthropomorphic robot in simulink environment. In: *International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, Alsace, France, and 21-23 July. (2015)
- [8] Santagati, C., Inzerillo, L., Di Paola, and F.: Image-based modeling techniques for architectural heritage 3d digitalization: Limits and potentialities. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 5 (2013).