Journal of Current Research in Engineering and Science
Bi-Annual Online Journal (ISSN : 2581 - 611X)

Volume 7- Issue 1, January 2024
Paper : 100

Implementing Simultaneous Localization and Mapping for Autonomous Navigation in ROS Turtlebot3

Surya Kodukula
Dept of Mechanical Engineering
GITAM (Deemed to be University)
Hyderabad, India
suryakodukula731@gmail.com

Vamshi Krishna Reddy Hanmanthgari
Dept of Mechanical Engineering
GITAM (Deemed to be University)
Hyderabad, India
vamshikrishna.h30@gmail.com

Siddhartha Gonella Venkata Saravana Krishna
Dept of Mechanical Engineering
GITAM (Deemed to be University)
Hyderabad, India
siddharthagvsk90@gmail.com

Kiran Kumar Abbili
Dept of Mechanical Engineering
GITAM (Deemed to be University)
Hyderabad, India
kabbili@gitam.edu

Abstract— ***This project presents an autonomous navigation system for a Turtlebot3 robot using the Robot Operating System (ROS). The system leverages the simultaneous localization and mapping (SLAM) algorithm to create a map of an unknown environment and then uses it to plan and execute autonomous navigation. The implementation includes both simulation in Gazebo and deployment on a physical Turtlebot3 robot. The SLAM algorithm used in this project is the gmapping package, which creates a 2D occupancy grid map of the environment using data from the Turtlebot3's laser range finder sensor. The resulting map is then used by the Turtlebot3's navigation stack, which plans and executes trajectories to move the robot autonomously towards a goal.***

The simulation in Gazebo was used to test and evaluate the system before deployment on the Turtlebot3 robot. The robot's autonomous navigation performance was tested in various scenarios, including navigating through narrow passages and avoiding obstacles.

The implementation on the physical Turtlebot3 robot involved setting up the hardware and software to communicate with ROS and deploying the SLAM and navigation stack packages. The robot was then able to autonomously navigate towards a goal while avoiding obstacles.

The results of this project demonstrate the feasibility and effectiveness of using ROS and the Turtlebot3 platform for autonomous navigation.

Keywords— ***Robot Operating System (ROS), SLAM, Gazebo, Navigation***

I. Introduction

Autonomous navigation, which involves the ability of a robot to plan and execute its own actions without human intervention, is a critical component of modern robotics. It enables robots to perform tasks in complex and dynamic environments, such as industrial automation, logistics, and search and rescue operations.

Autonomous navigation is a well-researched topic in the field of robotics, with numerous studies addressing various aspects of the problem. In this section, we provide an overview of the existing literature on autonomous navigation, with a focus on SLAM, path planning, and obstacle avoidance. SLAM is a fundamental technology for autonomous navigation that enables a robot to build a map of its environment and localize itself within it. SLAM can be achieved using different types of sensors, such as cameras, lidars, and sonars, and various algorithms, such as extended Kalman filter (EKF), particle filter (PF), and graph-based methods. One of the most popular SLAM algorithms is the simultaneous localization and mapping (SLAM), which uses a probabilistic framework to estimate the robot's pose and the map of the environment at the same time. SLAM has been extensively studied in the literature, with notable works including [1], [2], and [3].

Path planning is another critical component of autonomous navigation that involves finding an optimal path from the robot's current position to its goal while avoiding obstacles and other constraints. Path planning can be achieved using different techniques, such as potential fields, A*, RRT, and D* algorithms. One of the most widely used path planning algorithms is the A* algorithm, which uses a heuristic search to find the shortest path between two points. A* has been extensively studied in the literature, with notable works including [4], [5], and [6].

Obstacle avoidance is a key requirement for safe and efficient autonomous navigation, as it enables the robot to avoid collisions with static and dynamic obstacles in its environment. Obstacle avoidance can be achieved using different sensors, such as proximity sensors, cameras, and lidars, and various algorithms, such as potential fields, reactive methods, and model predictive control. One of the most effective obstacle avoidance methods is the reactive method, which uses a set of rules to steer the robot away from obstacles. Reactive methods have been extensively studied in the literature, with notable works including [7], [8], and [9].

While significant progress has been made in autonomous navigation, several challenges remain, such as robustness, scalability, and real-time performance.

The motive behind the project is to make a robot that could move autonomously in any environment to reach the final destination without the human intervention. In warehouses the objects or the items which are to be placed in particular location there were pallet trucks and unit carrier etc. here they require human where the same man power can be used for different place. So this gave us the motive to make compact model that can carry the weight on it and reach the destination.

II. METHODOLOGY

A. *Robot Operating System*

ROS is platform for robot software development where it gives us a set of packages to function the robot and it also gives us few simulation software where the working of robot in open world can be done in the system. It is a backbone for the robot to perform different operations in different circumstances.

The platform we have chosen here is the ROS Noetic robot . Which was the latest version supported for Ubuntu 20.04.5.

The following are the packages used and are listed below:

roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

The above package is used to move the robot.

roslaunch turtlebot3_slam turtlebot3_slam.launch

This package activates the RVIZ software which used to localize and position the object.

roslaunch turtlebot3_bringup turtlebot3_robot.launch

This package is used for interfacing the robot and the PC (the host) .

B. *Sensors And Equipment*

Fig.1 shows the robot mounted with LDS (Laser Distance Sensor), Raspberry Pi Module v2.1. The LDS senor has the capability of detection range from 160 – 8000 mm.

Once the equipment gets ready then, they are needed to be connected to the ROS . And the equipment consumes the low power . The setup does not take much time but the connections needed to be done carefully.

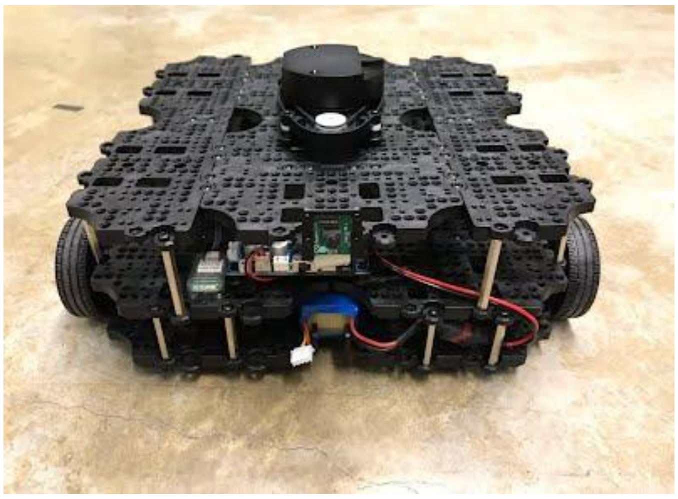


Fig.1. Turtlebot3 Waffle

C. *Mapping and Localization*

Once the whole equipment is connected then it is needed to be interfaced with the host PC and must be connect with the same network once it is done then using the slam package the visualization software open which marks the location of the things by LDS sensor within the inflation radius of 0.15m. Once mapping is done it locates the things and saves the objects over that point.

III. IMPLEMENTATION

The packages Teleop key and SLAM are to be installed, after installing them they must run in ubuntu. The commands are listed below:

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
roslaunch turtlebot3_slam turtlebot3_slam.launch
roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

Volume 7- Issue 1, January 2024
Paper : 84

The very last command helps to interface the robot to the monitor which when the instructions are given it will be followed.

Fig.1. Shows the autonomous navigation, the robot must map the environment by using the teleop key command which makes the robot move and SLAM command which generates the map of the environment. The generated map must be saved after that the robot must be given the orientation for the starting point and also the final destination to reach autonomously.

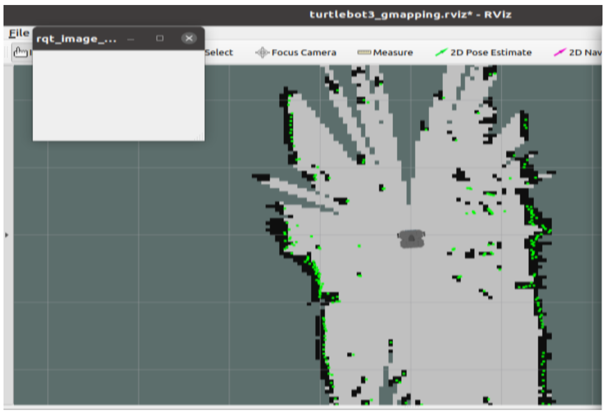
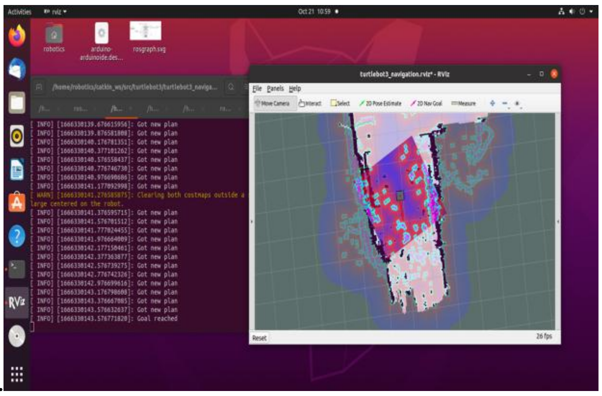


Fig. 2. Mapping the room using SLAM

The Fig.3 helps to understand the autonomous navigation visualization in RVIZ tool which in an environment it will be showing all the obstacles and the goal to be reached. As the obstacles being observed in the tool, it will be generating a new plan based on the actions it will reach to its final destination.

.

```
[ INFO] [1666330139.676615956]: Got new plan
[ INFO] [1666330139.876581000]: Got new plan
[ INFO] [1666330140.176781351]: Got new plan
[ INFO] [1666330140.377101262]: Got new plan
[ INFO] [1666330140.576558437]: Got new plan
[ INFO] [1666330140.776746730]: Got new plan
[ INFO] [1666330140.976690686]: Got new plan
[ INFO] [1666330141.177092998]: Got new plan
[ WARN] [1666330141.276585875]: Clearing both costmaps outside a
large centered on the robot.
[ INFO] [1666330141.376595715]: Got new plan
[ INFO] [1666330141.576781512]: Got new plan
[ INFO] [1666330141.777024455]: Got new plan
[ INFO] [1666330141.976664089]: Got new plan
[ INFO] [1666330142.177150461]: Got new plan
[ INFO] [1666330142.377363877]: Got new plan
[ INFO] [1666330142.576739275]: Got new plan
[ INFO] [1666330142.776742326]: Got new plan
[ INFO] [1666330142.976699616]: Got new plan
[ INFO] [1666330143.176798608]: Got new plan
[ INFO] [1666330143.376667085]: Got new plan
[ INFO] [1666330143.576632637]: Got new plan
[ INFO] [1666330143.576771820]: Goal reached
```

Fig. 3. Visualization of robot in RVIZ

IV. RESULTS AND DISCUSSION

Installing and testing the navigation stack through the simulation was straightforward execution as the instructions from site tutorials were clear. Testing the navigation stack with TurtleBot in a simulated world was also clear by following the tutorials.

Simply clicking the first Pose Estimate button on the TurtleBot and then clicking the Nav Goal button and the desired location in the map made the TurtleBot immediately auto-navigate to the destination. It is important to define the Pose Estimation directly on the robot and in the correct direction as failing to do so may cause problems with the navigation and in the end cause failing to find the destination. Problems with navigation can also occur when the destination is far away and the path is narrow. This will fail to create and find the path for the robot to navigate. The counter measurement for this purpose appeared to rotate recovery behaviour operation. The robot rotated twice around to scan the area and searched for a possible route. If the robot still failed to find a route it decided to abort and informed that it could not find valid control. The apparent solution was to give a destination point with smaller steps before giving the final goal of the destination. After giving the small steps first, the robot managed to find the final destination point. Martinez and Fernandez (2013) pointed out that the auto-navigation stack applies only to a certain type of robot for it to work. The navigation stack can only be implemented to a holonomic-wheeled with a different- trial drive. In addition, for the robot to observe the environment, a planar laser is needed to create localization and map. The first simulation test of auto-navigation was per- formed into a readymade map.

However, if a map is not readily available there is a solution based on mapping with the SLAM method. In the SLAM method, TurtleBot roams around the unknown area while sim- simultaneously scanning the area by using the robot's odometry and laser sensor data. The collected data is sent to the map server stack and mapping package in the ROS to form a map. The map creation based on SLAM in ROS with Gazebo simulation and Rviz visualization consisted of the following steps. First, the Simulation is started by ROS in Gazebo. Secondly, TurtleBot's movement and sensor measurements are created in Gazebo simulation and the results are exported to ROS. Thirdly, the calculation of SLAM mapping and robot localization is calculated in ROS. Finally, simulated data results are visualized and imported into Rviz from ROS[10].

When the mapping was tested, it was noticed that the tool had difficulties scanning the area when the TurtleBot roamed. The mapping status informed constantly that scan matching failed. Visually the problem was seen in the Rviz where the TurtleBot could not plot the scanned area and the location of the TurtleBot was also inaccurate as, occasionally, the TurtleBot made sudden teleportation

Volume 7- Issue 1, January 2024
Paper : 84

movements in the simulated world. To improve the ability to scan the area and also keep track of the TurtleBot's movement, the speed was decreased. This improved the performance and the chance for TurtleBot to create the area. Nevertheless, this did not entirely remove the difficulties in creating the map. As a result, the map scanning and creating process was a slow phase as each step had to be made sure that the TurtleBot was able to scan and register the area. Another way was to repeat the same route to create the map properly. The result of the created map was visualized through Rviz. The scanned area was successfully created if it appeared as a grey area in the Rviz. Through numerous trial and error methods, the map was created and the TurtleBot managed to auto-navigate around the created map.

V. CONCLUSIONS

In this research, we implemented an autonomous navigation system based on SLAM and ROS, using the Turtlebot3 platform and the Gazebo simulation environment. We used the Hector SLAM algorithm to build a map of the environment and the ROS navigation stack to enable the Turtlebot3 to navigate autonomously while avoiding obstacles. We conducted several experiments to test and evaluate the system's performance and found that it was robust, efficient, and accurate in various scenarios.

The results of this research demonstrate the effectiveness of SLAM-based autonomous navigation systems for mobile robots and highlight the potential of ROS as a framework for robotics research and development. The Turtlebot3 platform and the Gazebo simulation environment provided a convenient and cost-effective means for testing and evaluating the system's performance, enabling us to iterate and improve the system rapidly.

Future Scope

While the results of this research are promising, there is still room for improvement and further exploration. One area for future research is the integration of more advanced path planning and obstacle avoidance algorithms, such as MPC-based controllers or deep learning-based methods, which may enable the Turtlebot3 to navigate more complex environments with greater efficiency and safety.

Another area for future research is the integration of more sensors and hardware, such as LIDAR or RGB-D cameras, which may improve the accuracy and robustness of the SLAM algorithm and enable the robot to perceive its environment in greater detail.

REFERENCES

[1] Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. MIT press.

[2] Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part I. IEEE robotics & automation magazine, 13(2), 99-110.

[3] Stachniss, C., & Burgard, W. (2008). Information gain-based exploration using Rao-Blackwellized particle filters. Autonomous Robots, 25(3), 319-344.

[4] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics, (4), 100-107.

[5] LaValle, S. M. (2006). Planning algorithms. Cambridge university press.

[6] [6] Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. The International Journal of Robotics Research, 30(7), 846-894.

[7] [7] Borenstein, J., & Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots

[8] [8] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. The international journal of robotics research, 5(1), 90-98.

[9] [9] Vasquez, J. C., Broxham, J., Kolski, S., & Ramos, F. (2019). Model predictive control for obstacle avoidance in autonomous ground vehicles: A review. Annual Reviews in Control, 47, 146-162.

[10] [10] Afanasyev, I., Sagitov, A., Magid, E. (2015). ROS-Based SLAM for a Gazebo-Simulated Mobile Robot in Image-Based 3D Model of Indoor Environment. In: Battiato, S., Blanc-Talon, J., Gallo, G., Philips, W., Popescu, D., Scheunders, P. (eds) Advanced Concepts for Intelligent Vision Systems. ACIVS 2015. Lecture Notes in Computer Science(), vol 9386. Springer, Cham. https://doi.org/10.1007/978-3-319-25903-1_24.