ROS Simulation-Based Autonomous Navigation Systems and Object Detection

Swati Shinde, Tanvi Mahajan, Suyash Khachane, Saurabh Kulkarni, and Prasad Borle

Abstract Autonomous robots are becoming popular and are being used in many industries, due to their autonomy features. They are just like humans who have the ability to make decisions on their own without any human help. As the need for such robots is increasing, our paper aims to present an ROS autonomous navigation software system for autonomous robots, which is capable of creating 2D and 3D maps of the Simulation environment, localizing the robot in that environment and further performing path planning of the robot along with object detection using ROS. Moreover, various algorithms used for creating maps along with detailed internal working of the packages used for path planning are being discussed in this paper.

Keywords ROS · Turtlebot 2 · Autonomous navigation · Gmappinga · RTAB-map · Object detection

1 Introduction

Nowadays, robots are becoming more and more popular. Many industries have also started adopting robots for their quotidian works. Robots can be of various types like humanoid robots, teleoperated robots, autonomous robots. Each robot is used for different purposes. Autonomous robots, also called Autobots, perform tasks on their own without any human help. They just sense the environment with the help of sensors like cameras, lasers, infrared sensors, and then this sensed information is been processed which further helps them in navigating, avoiding obstacles or performing some specific tasks on their own. Thus, this paper mainly focuses on autonomous robots and aims to explain simulation of autonomous navigation on ROS robot Turtlebot 2. The autonomous navigation system has been integrated with ROS. This system is capable of creating 2D and 3D maps of the environment using SLAM algorithms and also performs object detection and navigation.

S. Shinde (✉) · T. Mahajan · S. Khachane · S. Kulkarni · P. Borle
Pimpri Chinchwad College of Engineering, Pune, Maharashtra 411035, India
e-mail: swaatti.shinde@gmail.com

D. P. Agrawal et al. (eds.), *Cyber Security, Privacy and Networking*, Lecture Notes in Networks and Systems 370, https://doi.org/10.1007/978-981-16-8664-1_4

Firstly, the paper explains the robot and the environment used for the simulation, the software tools and platform used. Secondly, the ROS autonomous navigation system has been elaborated, followed by object detection in the environment. Along with this, the results are evaluated.

2 Related Work

In [1], the authors have presented autonomous navigation of robot using Gmapping algorithm with the help of ROS. The robot was simulated on a Gazebo environment where SLAM and navigation were performed. Results say that the map of environment was successfully created, and the robot was able to perform navigation without colliding [2].

In research paper [3], authors have built an autonomous navigation platform [3] where the robot is able to create maps of environment and further navigate. They have designed this for indoor applications. Gmapping algorithm was used for the mapping process, and they have also evaluated the results with respect to mapping, localization and navigation. Their study says that the robot gives good response time [3] and reasonable time [3] to navigate from one point to another.

3 Robot and Environment

Turtlebot 2 which is the second generation of Turtlebot robot [4] is been for testing the algorithms for mapping and object detection in a simulated environment. It is a low-cost mobile robot used for education and research [4]. It contains a kukobi base, kinect mounting hardware, asus xtion pro live, etc. (Fig. 1).

Fig. 1 Turtlebot 2 [2]

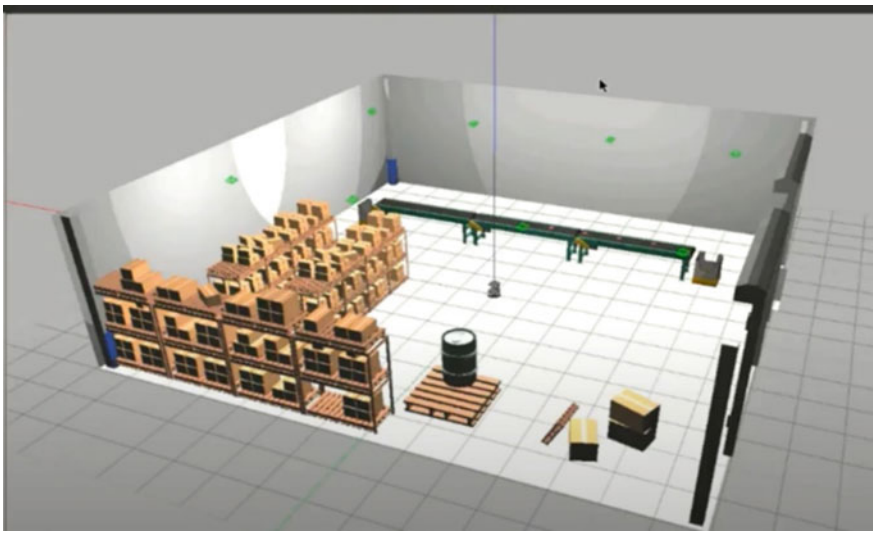

Fig. 2 Warehouse environment

Simulation environment is just a virtually created environment which can be used for testing of any application. The warehouse environment which consists of a conveyor belt, some wooden blocks and wooden storage area has been provided by RDS as shown in Fig. 2. This environment is being used for testing the robot and algorithms for mapping and navigation.

4 Software and Platforms

4.1 ROS

ROS is a robot operating system [5] is used as a middleware which provides libraries and tools to create robot applications. Running processes on ROS take place in the form of nodes, which may receive, post and multiplex sensor data, control messages passing between different nodes. ROS is an open-source platform [5] that supports various operating systems, but it is not a real operating system. It provides licensed packages, tools and libraries that are used to implement functionality of robot models, hardware drivers, planning, simulation tools and slam algorithms.

4.2 RDS

RDS is a platform provided by the construct. It helps programmers to program their robots, test the programs in real time [6] and simulate environments like warehouse environments, empty environments and golf environments. It also provides graphical tools like Rviz and rqt and robots like Turtlebot 2, Husky and R2C.

4.3 RVIZ

Rviz is a 3D visualization tool for the ROS framework. It helps us to view our robot and display robot's sensor data like laser scans, images captured by camera and Imu data. Maps created by the robot are visualized in Rviz. All the visualization processes, i.e. mapping, localization and navigation, are also visualized in Rviz.

5 ROS Autonomous Navigation

Robot navigation is about making a robot move autonomously in an environment. There are many techniques to make robots navigate. The most common is based on SLAM. Robot navigation [6] is elaborated in below steps:

1. Map Creation : Creating a map of the environment in which the robot will navigate.
2. Localization : It tells the current position of the robot in the map created.
3. Path Planning : To give a path for the robot to reach its goal position.

5.1 Map Creation

Robots must be familiar with the environment, and for this, the robot must be able to create a map of it. Maps are nothing but a representation of the environment where the robot will be navigating autonomously. SLAM helps achieve this task. SLAM is simultaneous localization and mapping. Within ROS, the most commonly used and widely accepted SLAM nodes are Gmapping & Hector SLAM. Some SLAM mapping algorithms are discussed below.

Gmapping. Gmapping is a ROS-based SLAM algorithm used for the creation of 2D maps of the environment. It takes inputs as the laser scanned data and odometry. ROS provides the slam_gmapping node [7] from the Gmapping package for mapping. This node subscribes to the laser scanned data, odometry, "/tf " topic and publishes "/map" topic. Once all parameters are tuned of slam_gmapping node, the map can be visualized in the rviz tool.

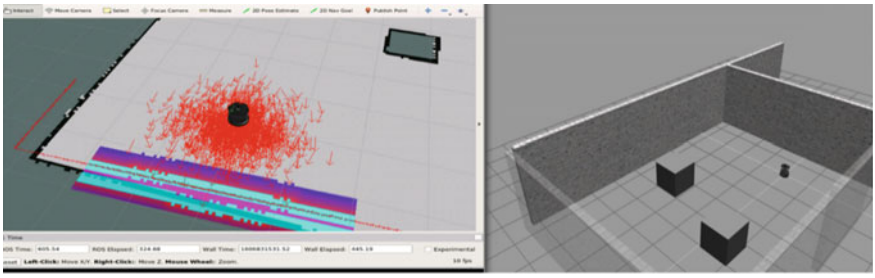

Fig. 3 Localization of robot

Hector Slam. Hector SLAM [8] takes advantage of the high update rate of laser scanners to provide an accurate estimate of the robot's location and pose while mapping the environment [8]. Hector SLAM does not require odometry for creating a map of the environment. It is potential enough to build a map using only laser scanned data.

RTAB-Map. RTAB-map is Real-Time Appearance-Based Mapping [9], a 3D slam algorithm used for creating 3D maps of the environment. RTAB-map takes laser scanned data along with odometry information to frame pose. Loop closure detection [10] ensures that the robot has visited that place or not and adds a new constraint to the map's graph. Graph optimizer is used to minimize error in the map. With generation 3D point clouds of environment RTAB-map 3D map.

5.2 Localization

An autonomous robot should know the environment and where it is in the environment. Localization means finding out the location or position of a robot in a given environment provided he has visited the environment at least once. ROS package AMCL helps localize a robot in the environment.

AMCL is an adaptive Monte Carlo localization approach, a probabilistic localization system for a robot moving in a 2D environment [11]. It takes input from the laser scan data, /tf topic, initial position of the robot, and most important is the 2D map. It publishes the robot's estimated pose on the map. In the rviz tool, we can visualize this and we can localize the robot in the rviz tool using 2D pose estimation by just specifying the position and direction of the robot facing towards.

In Fig. 3, we can see that the red arrows are indicating that the robot is facing towards the wall. This is the localization of Turtlebot 2 using AMCL.

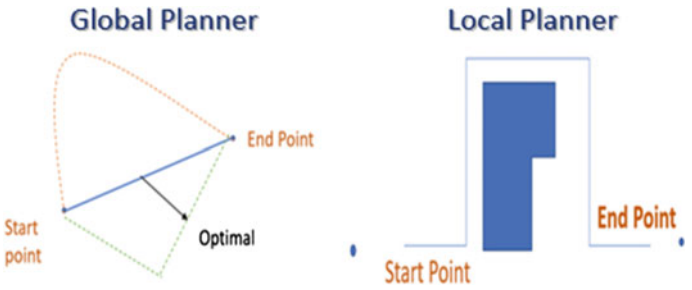

Fig. 4 Global planer and local planner

5.3 Path Planning

Path planning is nothing but it determines the sequence of steps the robot must take from current position to give goal or desired position. While planning the path, it takes into consideration the obstacles which are in the path. ROS has a package named move_base which helps to plan the path. This package contains the move_base node which acts as the path planner. Move_base node creates the costmaps based on the given map as input. Navigation is performed with the help move_base package which uses the concept of global planner and local planner.

Costmaps. Costmaps are the places which are safe for the robot. There are two costmaps—local costmaps and global costmaps. Local costmaps take the dynamic maps which take the sensor data reading into consideration which is used for getting the current motion of the robot, while the global costmaps are static maps which are static.

Global Planner. Global planner is used to calculate a path. It calculates a safe path for the robot based on the static map of the environment provided. Global planner uses the global costmap. When a goal is sent to the robot the move_base sends that goal to the global planner. Based on that goal, a path is calculated. For calculating paths, there are three global planners available : global_planner, Navfn [12]*, carrot_planner. Navfn is the most commonly used global planner. It internally uses the Djikstra algorithm.*

Local Planner. Local planner uses the local costmaps. When the path is calculated by a global planner, it is sent to the local planner. Local planner, based on the path given by global planner, sends commands to robots on /cmd_vel topic to move to the goal position. While approaching the goal position, it takes runtime sensor reading data into consideration, which means if any runtime obstacle comes into the environment which was not detected in the global costmap as while building the map it wasn't present, that is been detected in the local costmap. If any obstacle is detected by the local planner, then it recalculates the path for moving towards the goal. To achieve this, we need to tune parameters files of the move_base package.

Three basic parameter files are there—the global parameters file, local parameter file and common parameters file. All these are " .yaml" files (Fig. 4).

6 Object Detection

During navigation of the robot, obstacles in the robot's path can be handled by avoiding it using object detection results. YOLO ROS: real-time object detection for ROS, provides darkent_ros [13] a ROS-based packet for object detection for robots. DarkNet is an open source, fast, accurate neural network framework used with YOLOv3 [14] for object detection as it provides higher speed due to GPU computations. YOLOv3 is the real-time object detection technique which uses trained image weights [15] for detection of new objects which is provided by this package. This will take the camera topic as input, and it will publish the image detected, where the detected object will be bound by boxes. Moreover, it will also give us the accuracy match percentage of the objects detected.

7 Results

7.1 Room Map Creation

The map creation was performed on the warehouse environment, Fig. 5 shows a 3D map created, where the actual objects, walls and things placed in the environment are seen in the map using the RTAB-map algorithm, and Fig. 6 shows the 2D map created of the environment with Gmapping SLAM algorithm. From results, we can see that accurate maps were been built. Hence, Gmapping and RTAB-map have performed well in the warehouse environment with Turtlebot 2.

7.2 Object Detection

We have performed object detection in the object placed environment as shown in Fig. 7. This environment is created in gazebo, for the purpose of object detection where various objects like sofa, stop sign and laptop are been placed. With the help darknet_ros package,the person, traffic lights, stop sign, sofa and chair are being detected by Turtlebot 2 as shown in Fig. 8. Sofa was been detected with an 100% accuracy match, traffic light and stop sign with an accuracy match of 98%. Therefore, darknet_ros package has performed very well with Turtlebot 2 and object placed environment.

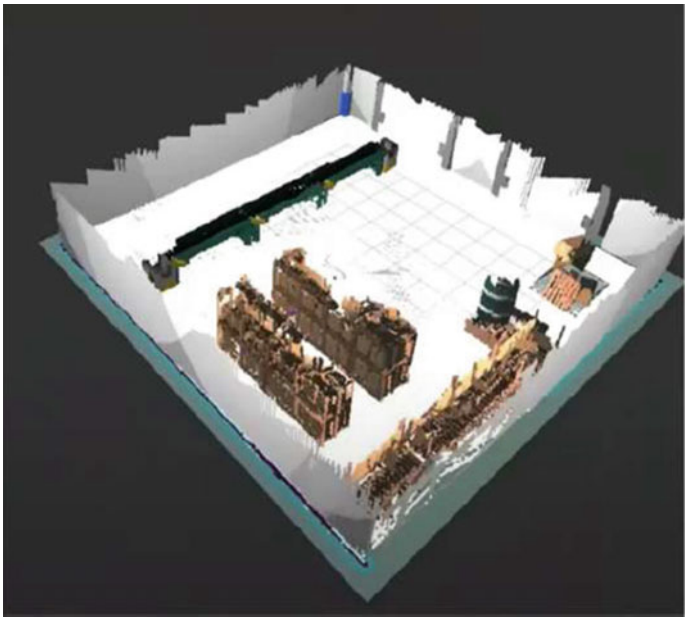

Fig. 5 RTAB-Map

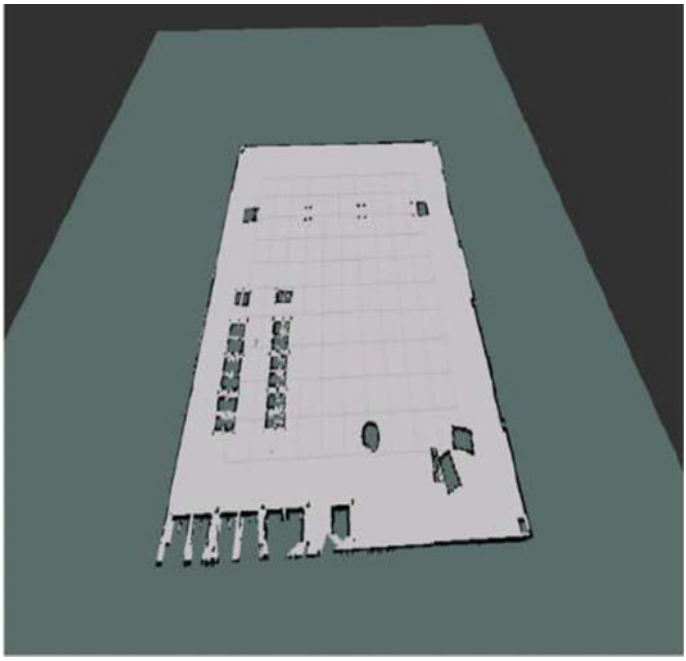

Fig. 6 Gmapping map

Fig. 7 Environment of object

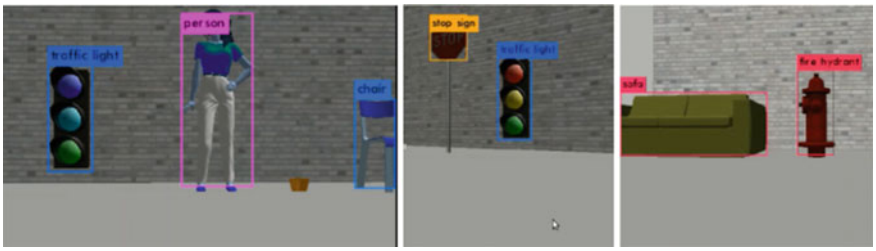

Fig. 8 Object detected in environment

7.3 Navigation

Turtlebot 2 has performed 2D navigation on the warehouse environment. The map created using the Gmapping algorithm as shown in, i.e. Fig. 6, was passed to the navigation system. With the help of AMCL package, the robot was able to localize itself in the environment. After giving a goal position to the robot, it moves towards the goal position with the help of global and local planner. The green line in Fig. 9 indicates the path of the robot to move to the desired goal. Robot was able to move to the desired goal position in good interval time.

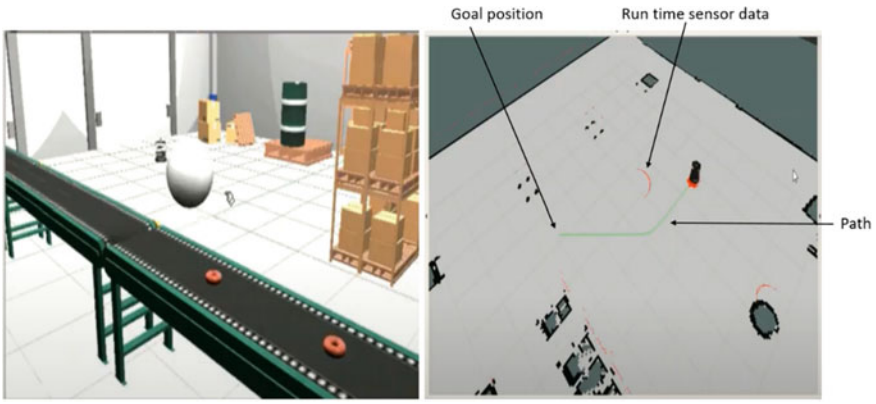

Fig. 9 Robot Navigation

8 Conclusion and Further Work

In this paper, we have presented the navigation software system for Turtlebot 2, where 2D maps using Gmapping and 3D maps using RTAB-map were built. Further, Turtlebot 2 was able to navigate in the known warehouse environment when a desired goal was given, with the help of ROS move_base package. Along with this, object detection was also performed on an object placed environment, where darknet_ros package was used. Turtlebot 2 was successfully able to detect the objects that were placed in that environment. Moreover, this study can be taken further where the warehouse environment and the object placed environment can be integrated where creating 3D maps along with object detection can take place at the same time.

References

1. Sumegh T, Mihir P, Pranjali, T Pratik K (2020) ROS based navigation using Turtlebot. ITM Web of Conferences, vol 32, p 01011
2. Thale SP, Prabhu MM, Thakur PV, Kadam P (2020) ROS based SLAM implementation for autonomous navigation using Turtlebot. In: ITM Web Conference, vol 32, p 01011
3. Megalingam RK, Chinta R, Sreekant S, Raj A (2019) ROS based autonomous indoor navigation simulation using SLAM algorithm
4. Turtlebot2, https://www.turtlebot.com/turtlebot2/
5. ROS, http://wiki.ros.org/ROS/Introduction
6. The ROS development studio by the construct. https://www.theconstructsim.com/the-ros-development-studio-by-the-construct/
7. Lin Q, Ke Z, Bi S, Xu S, Liang Y, Hong F, Feng L (2017) Indoor mapping using gmapping on embedded systems. In: IEEE International conference on robotics and biomimetics (ROBIO), pp 2444–2449

8. Yu N, Zhang B (2018) An improved hector SLAM algorithm based on information fusion for mobile robot. In: 2018 5th IEEE International conference on cloud computing and intelligence systems (CCIS)
9. Das S (2018) Simultaneous localization and mapping (SLAM) using RTAB-MAP. Int J Sci Eng Res 9(8)
10. Angeli A, Doncieux S, Meyer J, Filliat D (2008) Real-time visual loop-closure detection. In: 2008 IEEE International conference on robotics and automation
11. ROS AMCL. http://wiki.ros.org/amcl
12. Zheng K (2019) ROS navigation tuning guide. In: IEEE International conference on robotics and automation 8 Apr 2019
13. datknet_ros. http://wiki.ros.org/darknet_ros
14. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. In: Computer vision and pattern recognition. IEEE, 8 Apr 2018
15. Redmon J (2017) Darknet: open source neural networks in C. https://pjreddie.com/darknet/