World Academy of Science, Engineering and Technology
International Journal of Mechanical and Mechatronics Engineering
Vol:15, No:3, 2021

Robot Operating System-Based SLAM for a Gazebo-Simulated Turtlebot2 in 2d Indoor Environment with Cartographer Algorithm

Wilayat Ali, Li Sheng, Waleed Ahmed

***Abstract*—**The ability of the robot to make simultaneously map of the environment and localize itself with respect to that environment is the most important element of mobile robots. To solve SLAM many algorithms could be utilized to build up the SLAM process and SLAM is a developing area in Robotics research. Robot Operating System (ROS) is one of the frameworks which provide multiple algorithm nodes to work with and provide a transmission layer to robots. Manyof these algorithms extensively in use are Hector SLAM, Gmapping and Cartographer SLAM. This paper describes a ROS-based Simultaneous localization and mapping (SLAM) library Google Cartographer mapping, which is open-source algorithm. The algorithm was applied to create a map using laser and pose data from 2d Lidar that was placed on a mobile robot. The model robot uses the gazebo package and simulated in Rviz. Our research work's primary goal is to obtain mapping through Cartographer SLAM algorithm in a static indoor environment. From our research, it is shown that for indoor environments cartographer is an applicable algorithm to generate 2d maps with LIDAR placed on mobile robot because it uses both odometry and poses estimation. The algorithm has been evaluated and maps are constructed against the SLAM algorithms presented by Turtlebot2 in the static indoor environment.

***Keywords*—**SLAM, ROS, navigation, localization and mapping, Gazebo, Rviz, Turtlebot2, SLAM algorithms, 2d Indoor environment, Cartographer.

I. Introduction

THE ROS has encouraged solutions for various robotics problems in the form of ROS wrappers/packages. SLAM is one of these problems, a problem resolved by evaluating the robot position and a map of its operating environment at the same time. Various methods try to solve the SLAM problem, which uses a prediction-correction approach to evaluate current robot position and map and can be divided into different groups depending on a set of sensors they use and details of their performance (e.g., a genre of the map, aspect of usage, etc.).

With the innovation in the sensing and robotics technology, the problem known as SLAM turned out to be one of the most explored problems by the robotic research community in the past decade, which has produced various contributions [1], [2]. There are various methods to solve the SLAM problem; however, it is difficult to choose the best algorithm among several algorithms. We have to compare a few methods to get better results to find the best method among different algorithms. The ROS was presented as a flexible framework for robot software [3]. The ROS philosophy was to set up a framework integrated into various robotics platforms by forming little changes in a code. Additionally, ROS allows researchers an option to quick and easy simulation and real-world experiments. The majority of the SLAM libraries support ROS API. For our research, we compare the most commonly used 2d Lidar based SLAM libraries that possess ROS wrappers – Hector mapping [4], Google Cartographer [5] and Gmapping [6] for the indoor static environment. In this research, the author proposed indoor mapping using kinetic and ROS, which found that the kinetic sensor produces a more precise map than other laser ranger finder data [7].

Now, after getting a map, the next important thing is to evaluate the results. SLAM techniques comparison is made by different works. Like in work presented, [8] investigated several SLAM methods, presuming the comparative analysis of all trajectories obtained by using different sensor data (conventional camera, LIDAR, stereo ZED camera, and Kinect depth sensor) during the experiment with UGV prototype motion in the indoor environment [8], [9]. To get results with the odometry process, [10] determines a comparative analysis of a quadrotor UAV trajectories evaluated by processing integrated sensors (camera and IMU) with ROS-based monocular visual odometry.

In [11], the author checked the efficiency of SLAM based robot model under ROS by calculating the time taken by the robot model to reach the destination point. In this work, the author proposed an experimental analysis of two ROS compatible SLAM packages: Gmapping and RTAB-Map with Gazebo simulation founding that for ground robots, the RTAB-Map is a more suitable solution [12]. In the research presented by [13] comparison of several SLAM methods (Gmapping, Hector SLAM and Karto SLAM) are implemented in ROS and it is verified that Gmapping had the best results when the laser range finder sensor is added. In [14], several laser-based 2D SLAM techniques based on the ROS are conducted. All the approaches have been examined and compared in 2D simulations and real-world experiments. The experimental results collected were based on k-nearest neighbor's concept. In [15], the author presented the metrics based on the capture characteristics method to compare the SLAM algorithms to find the most accurate one. Reference [16] compares Lidar-based 2d SLAM maps by calculating the average distance to the nearest neighbor presented by the

Wilayat Ali is with the Nanjing University of Science and Technology, China (e-mail: wilayat512@njust.edu.cn).

FARO laser tracker in the static indoor environment.

The main purpose of our research is to get a better result of mapping by implementing Cartographer mapping to solve SLAM based robot model Turtlebot2 implemented in ROS and Gazebo in an indoor static environment.

The paper is organized as; Section I describes the introduction and the related work. The next section describes the methodology used in this paper. Section III covers the system setup and indoor environment creation in Gazebo is discussed. Section IV explains the experimental simulation results of the paper. Section V describes the conclusion and future work of this research.

II. Methodology

SLAM is a method used for mobile robots that lets them build a map and localize on the map simultaneously. SLAM enables the robot to achieve this task by knowing how the environment looks (mapping) and staying in the environment (localization). SLAM uses different sensors to collect data and build a map of the surrounding. ROS can implement various SLAM algorithms such as Gmapping, Hector, and Google Cartographer mapping.

A. Full SLAM

$$p(x_{0:t}, m \mid z_{1:t}, u_{1:t}) \tag{1}$$

The full SLAM estimates entire path and map in (1). Online SLAM:

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \iint \dots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) dx_1 dx_2 \dots dx_{t-1}$$

The online SALM estimates most recent pose and map

Feature-Based SLAM

In the featured-based SLAM given data of the Robot, the robot's control is

$$U_{1:k} = \{u_1, u_2, \dots, u_k\}$$

The relative observations to the Robot in featured-based SLAM is

$$Z_{1:k} = \{z_1, z_2, \dots, z_k\}$$

Wanted Map and Path of the Robot

The map feature of the Robot is:

$$m = \{m_1, m_2, \dots, m_n\}$$

The path of the Robot in featured-based SLAM is

$$X_{1:k} = \{x_1, x_2, \dots, x_k\}$$

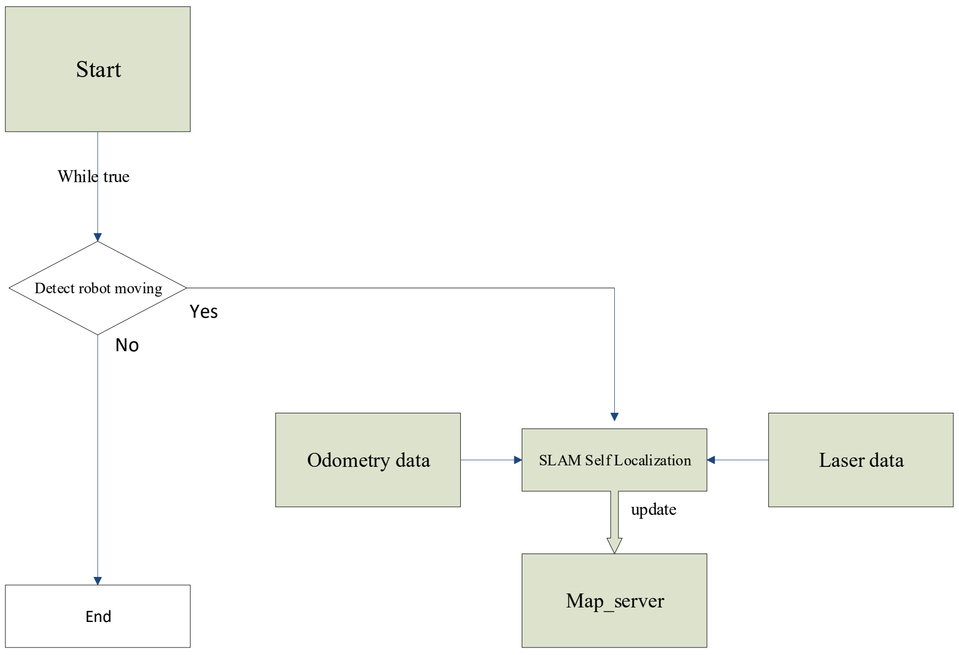


Fig. 1 SLAM Framework Overview

B. Google Cartographer

Cartographer is an open-source project established in 2016. It can generate 2D and 3D real-time SLAM in a various-platform with sensor configuration. The principle of cartographer is set up on graph optimization. The main idea is to eliminate combined errors generated by loop closure detection during the mapping process. The core unit for closed-loop detection is submap. A submap is made up of a defined number of laser scans. When a laser scan is placed into its corresponding submap, its optimal position in the submap is evaluated based on the submap's existing laser scan and sensor data. The increasing of errors in creating sub-maps in a small time is considered small enough. Although, as further sub-maps are created over time, the error extension between sub-maps would increase. Therefore, it is necessary to properly optimize the pose of these sub-maps through loop closure detection to eliminate these additional errors, which transforms the problem into a pose optimization problem. In this case, the SLAM problem is modified into a nonlinear least-squares problem. The optimal trajectory in the robot SLAM process can be shown as solving the robot pose so that the following error square function is minimized:

$$F(X) = \sum_{\langle i,j\in C\rangle} \underbrace{e(X_i, X_j, Z_{ij})^T \Omega_{ij} e(X_i, X_j, Z_{ij})}_{F_{ij}} \quad (2)$$
$$X^* = \underset{X}{arg\,min}\, F(X)$$

This system consists of expression and position of the robot, while the pose of the robot is enhanced by global and local optimization. The translation and rotation are described as first two variables. The matching of the lidar scanning frame and the sub-map are locally optimized. The global optimization part is to complete global map optimization according to the pose relationship between the scan frames after finding the closed-loop scan frame [19].

C. Local Optimization

Local optimization is the process of relating the lidar scanning frame with the sub-map, and iteratively aligning the lidar scanning frame and the sub-map reference frame to generate the sub-map. The lidar scan is frame, and is at origin (0,0), and the scan point is described as $H = \{h_k\}_{k=1,\mathrm{L}_s,k}, h_k \in R$, the pose transformation of the scan frame to the submap is expressed as, which can rigidly map the scan points from the scan frame to the submap.

$$T_\xi p = \underbrace{\begin{pmatrix} \cos\xi_\theta & -\sin\xi_\theta \\ \sin\xi_\theta & \cos\xi_\theta \end{pmatrix}}_{R_\xi} p + \underbrace{\begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}}_{t_\xi} \quad (3)$$

Various iterative scan frames generate a submap that takes the form of a probability grid with a resolution of r. For every grid point, we define a corresponding pixel. Every time a new scan is set up into the probability grid, a set of hit grid points and disjoint miss grid points are evaluated. For the hit point, we put the adjacent grid points into the hit set. For the miss points, collect all the relevant points on the scan center and scan point join rays to the miss set, except for the points in the hit set. For every point that has not been evaluated before, and if this has been observed before, then the point probability is updated according to equations.

$$\text{odds}\,(p) = \frac{p}{1-p}$$
$$M_{\text{new}}\,(x) = \text{clamp}\left(\text{odds}^{-1}\left(\text{odds}(M_{\text{old}}\,(x)) \cdot \text{odds}(p_{\text{hit}}\,)\right)\right)$$

Ahead of constructing the scan map into sub-map, the scan frame is built up with the current sub-map, and the k scan point maps are put by the nonlinear least square's optimization. To get matching better a k scan points hit point, the pose is transformed and the probability value in the sub-map will be paired and each point to be displayed and must be a big probability to be hit.

$$\underset{\xi}{\text{argmin}} \sum_{k=1}^{K} \left(1 - M_{\text{smooth}}\left(T_\xi h_k\right)\right)^2 \quad (4)$$

As the least square issue is a local optimal problem, a fine initial value of the pose has a good impact on the output. Therefore, the IMU may be utilized to produce a scan-matched rotation variable for the initial pose value.

D. Global Optimimization

Global optimization is carried out through closed loop detection. While each lidar scan frame is only matched with a submap containing some recent scan frames, the build error is slowly accumulated. To get rid of the accumulated error, a Sparse Pose Adjustment method is used to optimize all. The pose of the scan and submap is given in (5):

$$\underset{\Xi^m, \Xi^m}{\arg min} \frac{1}{2} \sum_{ij} \rho\left(E^2(\xi_i^m, \xi_j^i; \Sigma_{ij}\,, \xi_{ij})\right) \quad (5)$$

ρ is a loss function, and the purpose of using it is to reduce the impact of outliers added to the optimization problem on the system. $\Xi^m = \{\xi_i^m\}_{i=1,\downarrow,m}$ and $\Xi^s = \{\xi_j^s\}_{j=1,\mathrm{L},n}$ are the poses of submap and scan in the world coordinate system, respectively, and are optimized under the constraints of the given relative pose ξ_{ij} and the associated covariance matrix Σ_{ij} . For paired Submap *i* and scan *j*, ξ_{ij} represents the relative pose of scan in the matched. Submap coordinate system, and the covariance matrix $\sum_{ij}$ can be described by using ceres covariance estimation feature process. The submaps which are inserted from the pose of the lidar scanning will be stored in the memory. For closed-loop detection the sub-map and scan frame are taken for consideration. When a good closed-loop match is obtained it is added to the global optimization.

When constructing a submap is completed, the submap would be added to the loop closure detection and it takes into account all sub-maps that have been created. When a new laser scan is added to the map, if the laser scan's estimated pose is close to that of a laser scan of a map's submap, the closed-loop could be found through a specific scan match strategy. The cartographer's scan match strategy takes a window around the estimated pose of the newly added laser scan and then looks for a possible match for the laser scan in the window. If an appropriate match is found, it would add the matching closed-loop to the pose optimization problem. The cartographer's key point is to create sub-maps that fuse multi-sensor data and complete closed-loop detection based on scan matching [17], [18].

E. Robot Operating System

The ROS is one of the most popular free and open-source middleware for building robot programming and helps developers and researchers to build and reuse code between robotics applications. ROS contains different packages, drivers, libraries, and tools for developing and building robot applications. This research uses ROS as its primary foundation since it consists of the required features like SLAM algorithms and robotics control.

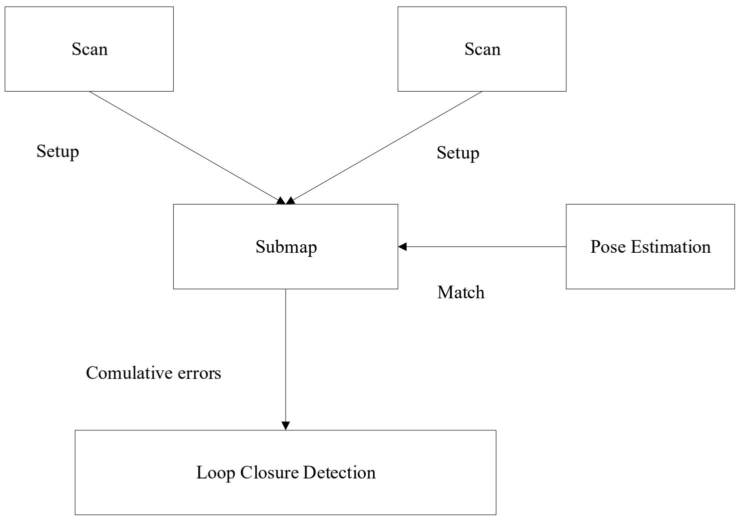


Fig. 2 Cartographer overview

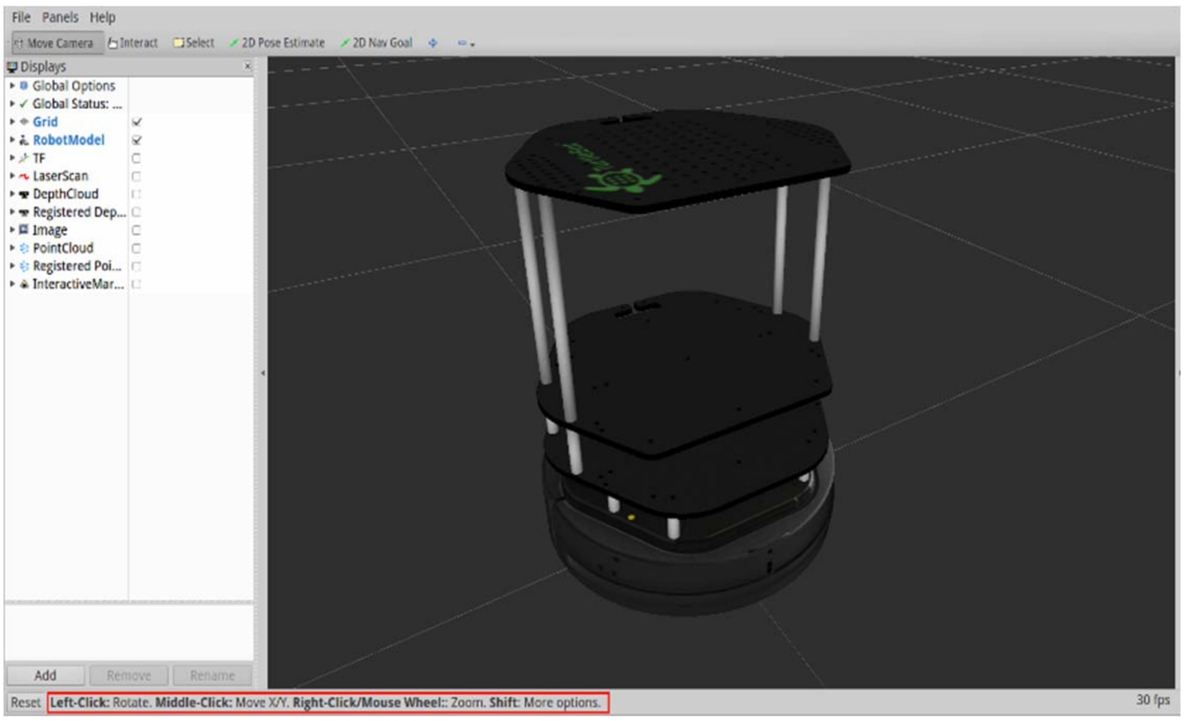


Fig. 3 Turtlebot2 in Rviz Environment

F. Turtlebto2

The platform for this research is decided as the Turtlebot2 having affordable cost with open-source software, which is in turn built upon the mobile base called kobuki with sharing wide sensor installed on it. Turtlebot2 conveniently provides the opportunity to test through a ROS simulation with the Gazebo simulator and RVIZ visualization tool.

G. Gazebo

The Gazebo collaborated to ROS is an open-source 3d robotic simulator that allows users to create complex indoor and outdoor environments and simulate the robot within the created environment. It comes up with a set of ROSs API that allows us to change and get information about several aspects of the simulated world.

H. Rviz

The Rviz is a 3D visualization simulator tool that allows us to visualize the sensor data in the 3D environment. For example, if we run a turtlebot2 robot model in Gazebo, which comes with a kinetic sensor, the kinetic sensor's laser scan value can be visualized in Rviz. From this laser scan data, the user can build a map for navigation, and we have access and graphically represent the values using laser scans and images in Rviz.

III. System Setup

A. Robot Simulation under ROS Gazebo

- ➢ ROS initiates simulation in Gazebo.
- ➢ Gazebo simulates the robot locomotion and sensor measurements.
- ➢ ROS imports the simulation results from Gazebo.
- ➢ ROS evaluates the robot localization and mapping SLAM.
- ➢ Rviz imports and visualizes the simulation data with real-time sensor data and robot localization and mapping.

B. Maps Construction from ROS-Based SLAM Algorithms

We have obtained the maps in the occupancy grid form and saved using the map_saver node using the map_server package.

C. Odometry

Odometry is the use of data from moving sensors to estimate the change in position over time. Odometry is used by some robots, whether legged or wheeled, to estimate their position relative to a starting location. The simulated TurtleBot will be used to demonstrate the odometry display possible in Rviz. The commands executed on the remote computer to start Gazebo for simulation and rviz for visualization and get the results.

IV. Experimental Results and Discussion

In this section, we show the examples of the availability and effectiveness of the SLAM system. The implementation process is carried out under the ROS, which is an open-source robot operating software. ROS is based on distributed structure, and nodes communicate with each other through the message mechanism and the data exchange is implemented through publishing or subscribing topics. In ROS, we can update the estimated robot coordinate position by the transform frame (tf) between the robot position coordinate (scanmatcher_frame) and the map coordinate (map_frame), which is stationary and usually express the starting point of scan matching process.

To obtain the simulation results of mapping first we need the simulation environment for the robot to move around. We have created the indoor static environment in Gazebo for this work in which the Turtlebot2 robot will move around and navigate and localized itself in that indoor static environment.

A. The Indoor Environment Created in Gazebo

To obtain the simulation results of mapping first we need the simulation environment for the robot to move around. We have created the indoor static environment in Gazebo. This research study's indoor static environment is created in Gazebo for the robot model imported Turtlebot2 to simulate and perform navigation in that environment. The sample of the Gazebo environment created is in Fig. 4.

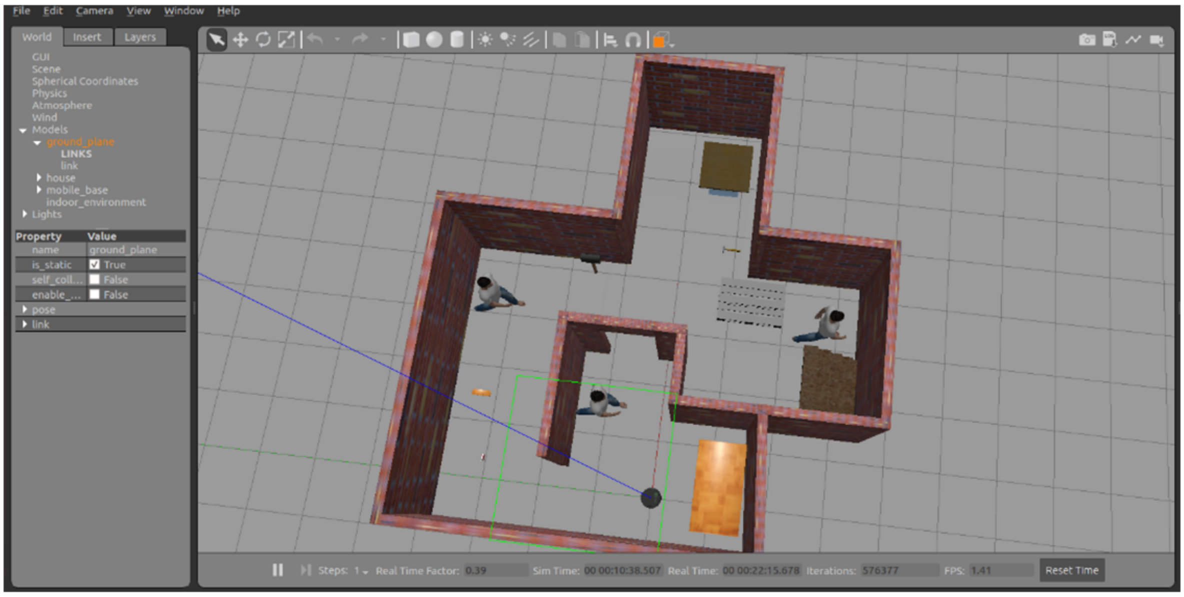
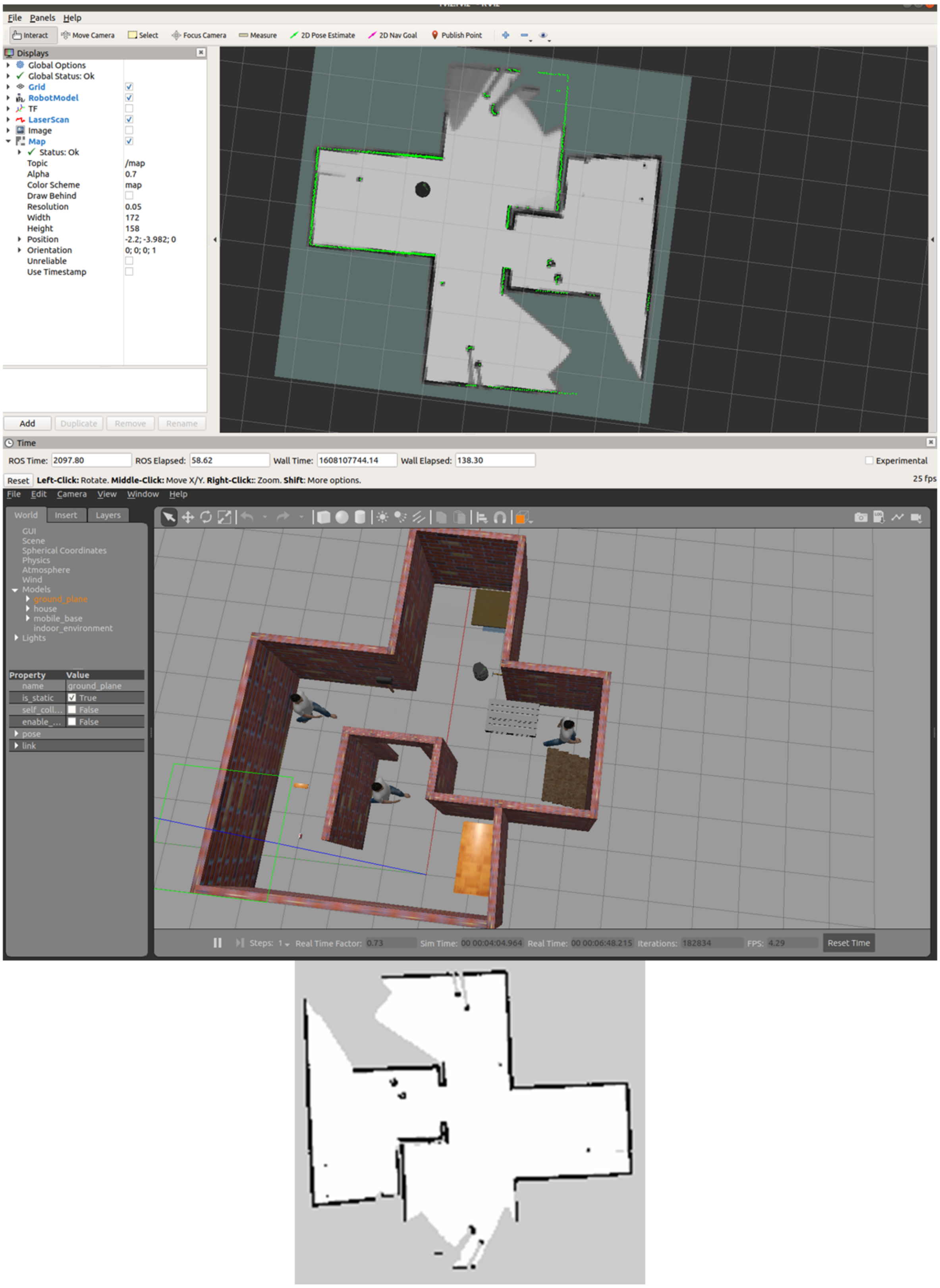


Fig. 4 Indoor Environment created for Simulation

B. Cartographer Results

The mapping results generated from Google Cartographer mapping are as Figs. 5 and 6 with robot moving around various stages. Figs. 5 and 6 show and explain the position of the robot, the Rviz mapping and the maps saved by the map_server package of ROS. The results obtained from our simulation are very clear with accordance to the time, obstacle avoidance, detection and alignment.



Global Options	
Global Status: Ok	
Grid	✓
RobotModel	✓
TF	
LaserScan	✓
Image	
Map	✓
Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	
Resolution	0.05
Width	172
Height	158
Position	-2.2; -3.982; 0
Orientation	0; 0; 0; 1
Unreliable	
Use Timestamp	

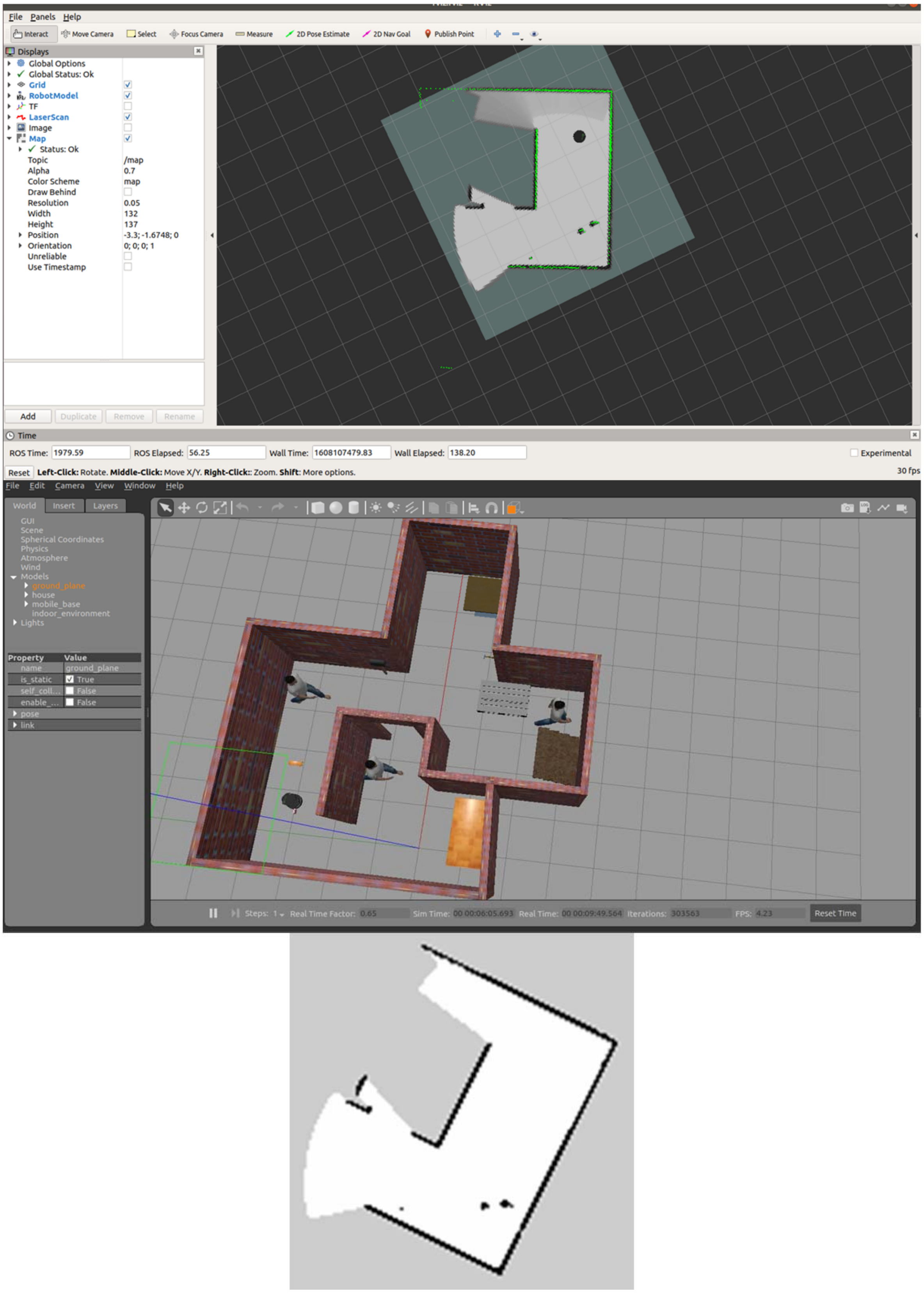
ROS Time: 2097.80 ROS Elapsed: 58.62 Wall Time: 1608107744.14 Wall Elapsed: 138.30

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift:** More options.

Property	Value
name	ground_plane
is_static	✓ True
self_coll...	False
enable_...	False
pose	
link	

Steps: 1 Real Time Factor: 0.73 Sim Time: 00 00:04:04.964 Real Time: 00 00:06:48.215 Iterations: 182834 FPS: 4.29

Fig. 5 Mapping result along with Gazebo and Rviz



Global Options
✓ Global Status: Ok
Grid ✓
RobotModel ✓
TF
LaserScan ✓
Image
Map ✓
✓ Status: Ok

Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	
Resolution	0.05
Width	132
Height	137
Position	-3.3; -1.6748; 0
Orientation	0; 0; 0; 1
Unreliable	
Use Timestamp	

ROS Time: 1979.59 ROS Elapsed: 56.25 Wall Time: 1608107479.83 Wall Elapsed: 138.20

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click::** Zoom. **Shift**: More options.

Property	Value
name	ground_plane
is_static	✓ True
self_coll...	False
enable_...	False
pose	
link	

Steps: 1 Real Time Factor: 0.65 Sim Time: 00 00:06:05.693 Real Time: 00 00:09:49.564 Iterations: 303563 FPS: 4.23

Fig. 6 Mapping results at other stage along with Gazebo and Rviz

C. Cartographer Active Nodes

Figs. 7 and 8 describe the active nodes of cartographer. It is shown how the cartographer is performing; for example, how the data originate from odometry to gazebo and then to Rviz to perform mapping.

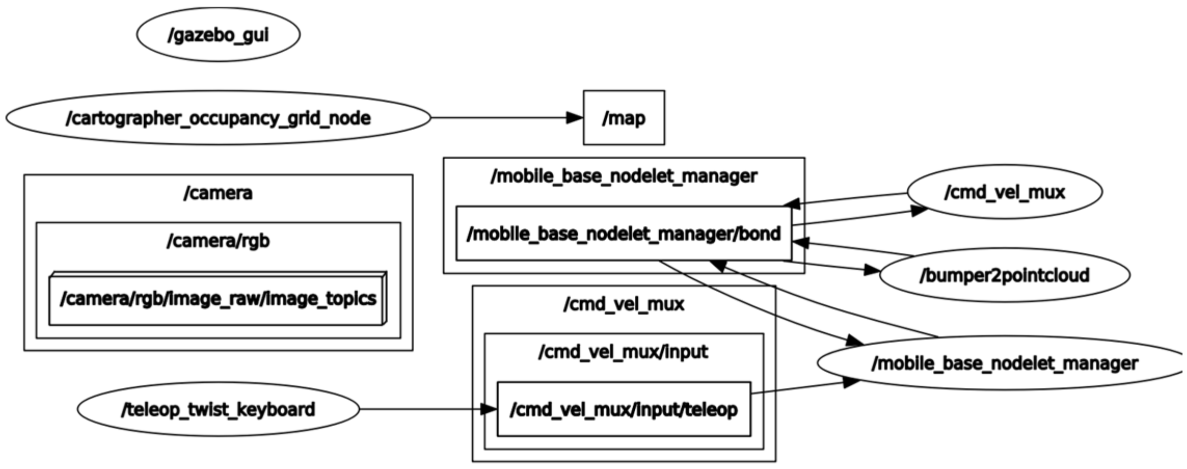

/gazebo_gui
/cartographer_occupancy_grid_node
/map
/camera
/camera/rgb
/camera/rgb/image_raw/image_topics
/mobile_base_nodelet_manager
/mobile_base_nodelet_manager/bond
/cmd_vel_mux
/bumper2pointcloud
/cmd_vel_mux
/cmd_vel_mux/input
/mobile_base_nodelet_manager
/teleop_twist_keyboard
/cmd_vel_mux/input/teleop

Fig. 7 Active nodes of Cartographer

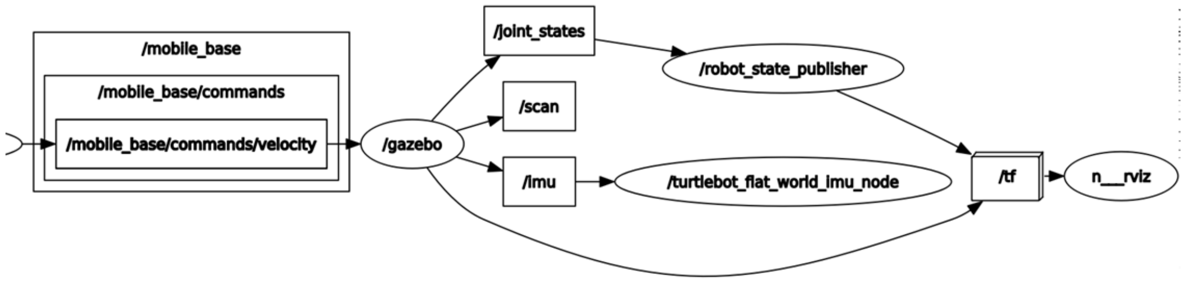


Fig. 8 Active nodes of Cartographer Algorithm

V. Conclusion and Future Work

In this paper, we have analyzed and evaluated the ROS based SLAM framework using turtlebot2 in an indoor static environment. We have implemented the Cartographer algorithm which is an open-source library. The results we obtained from the simulation in ROS-Gazebo were impressive with respect to alignment, time and obstacle avoidance. For the future aspects, the comparison research study shown in this research must not be only for 2D maps but 3D mapping. A 3D map can be created through Octomap. The next step is to extend the following research with 3D SLAM maps with Octomap. We could perform the mapping through these algorithms in real-world experiments with a personal turtlebot2 robot's real kit to perform navigation for future reference.

References

[1] C. Stachniss, J. J. Leonard, and S. Thrun, "Simultaneous localization and mapping," in *Springer Handbook of Robotics*: Springer, 2016, pp. 1153-1176.

[2] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," vol. 32, no. 6, pp. 1309-1332, 2016.

[3] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3, no. 3.2, p. 5: Kobe, Japan.

[4] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*, 2011, pp. 155-160: IEEE.

[5] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271-1278: IEEE.

[6] G. Grisetti, C. Stachniss, and W. J. I. t. o. R. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," vol. 23, no. 1, pp. 34-46, 2007.

[7] H. I. M. A. Omara and K. S. M. Sahari, "Indoor mapping using kinect and ROS," in *2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR)*, 2015, pp. 110-116: IEEE.

[8] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," in *2017 14th Workshop on Positioning, Navigation and Communications (WPNC)*, 2017, pp. 1-6: IEEE.

[9] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," in *2018 International Conference on Intelligent Systems (IS)*, 2018, pp. 400-407: IEEE.

[10] A. Gabdullin, G. Shvedov, M. Ivanou, and I. Afanasyev, "Analysis of onboard sensor-based odometry for a quadrotor uav in outdoor environment," in *Int. Conf. on Artif. Life and Robotics (ICAROB)*, 2018.

[11] R. K. Megalingam, C. R. Teja, S. Sreekanth, and A. J. I. J. P. A. M. Raj, "ROS based autonomous indoor navigation simulation using SLAM algorithm," vol. 118, no. 7, pp. 199-205, 2018.

[12] B. M. da Silva, R. S. Xavier, and L. M. Gonçalves, "Mapping and Navigation for Indoor Robots under ROS: An Experimental Analysis," 2019.

[13] F. Duchoň *et al.*, "Verification of SLAM Methods Implemented in ROS," 2019.

[14] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in robot operating system," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1-6: IEEE.

[15] A. Filatov, A. Filatov, K. Krinkin, B. Chen, and D. Molodan, "2d slam quality evaluation methods," in *2017 21st Conference of Open Innovations Association (FRUCT)*, 2017, pp. 120-126: IEEE.

[16] R. Yagfarov, M. Ivanou, and I. Afanasyev, "Map comparison of lidar-based 2d slam algorithms using precise ground truth," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 1979-1983: IEEE.

[17] M. Abouzahir, A. Elouardi, R. Latif, S. Bouaziz, A. J. R. Tajer, and A. Systems, "Embedding SLAM algorithms: Has it come of age?," vol. 100, pp. 14-26, 2018.

[18] R. Munguia, B. Castillo-Toledo, and A. J. S. Grau, "A robust approach for a filter-based monocular simultaneous localization and mapping (SLAM) system," vol. 13, no. 7, pp. 8501-8522, 2013.

[19] C. Zhi and S. Xiumin, "Research on Cartographer Algorithm based on Low Cost Lidar."