A Study of ROS Gmapping SLAM Package and its Accuracy Under Different Lidar Noises

Abstract

Simultaneous Localization and Mapping is an important development that is being applied in many areas ranging from self-cleaning vacuums at homes, to autonomous vehicles, and mobile robots. The SLAM problem has been studied for several years, and many different strategies have been developed to effectively perform it. Robot Operating System (ROS) has made SLAM very accessible, and easy to approach. In this paper, one of the best, and commonly used SLAM packages in ROS; Gmapping [1], was first briefly studied. Simulations using a turtlebot waffle were then performed in a virtual Gazebo environment, to evaluate the accuracy with which Gmapping measures the positions of certain critical points within the environment, under different gaussian lidar noises. It was found that the algorithm performed well with maximum errors being approximately around 0.1 m, when the gaussian lidar noises had a standard deviation between 0 - 0.05 m. However, the errors became more significant, ranging between approximately 0.3 - 0.75 m when the standard deviation of the lidar noise was increased to 0.1 m.

Introduction

Simultaneous Localization and Mapping is the process by which a mobile robot can incrementally build the map of an environment while simultaneously localizing itself within that map. The map generated by SLAM is used for autonomous navigation of the robot [2], making it critical to autonomous robots [3].

SLAM has been an area of research for many decades, and several methods have been developed to perform it. One of the mainly used technique is the Extended Kalman Filter (EKF) [4]. The Kalman Filter is a Bayes filter which assumes a Gaussian state and linear Gaussian observation. The Extended Kalman Filter is an extended version of the Kalman Filter which can be used in non-linear systems as well. Here, the non-linear systems are linearized about the estimated current mean and covariance [4]. Particle filters were later introduced since they could also deal with non-Gaussian distributions, unlike the EFK [4]. Another approach was to use a graph-based SLAM which utilizes graph to solve the problem. Here each pose of the robot during mapping is represented by a node in the graph, and the most likely orientation of the poses are generated from the data presented as edges [5].

The Robot Operating System (ROS) is the most popular and developed robotic framework for robotic applications nowadays [6] [7]. ROS allows researchers to perform simulations as well as real world experiments with the help of gazebo and rviz. In [7], 5 different types of laser-based 2D SLAM techniques that are available on ROS; HectorSLAM, Gmapping, KartoSLAM, CoreSLAM, and LagoSLAM were evaluated, and compared in 2D simulations, and real-world experiments. It was found that Gmapping produced the best results, while maintaining a low CPU load, making it very robust.

Gmapping SLAM

Gmapping SLAM uses an improved version of RBPF (Rao-Blackwellized Particle Filters) for grid mapping [8]. This algorithm was introduced in [9] and is summarized as follows.

RBPF works by estimating the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the map m and the trajectory $x_{1:t} = x_1, \ldots, x_t$ of the robot, given the observations $z_{1:t} = z_1, \ldots, z_t$ and the odometry measurements $u_{1:t-1} = u_1, \ldots, u_{t-1}$ obtained by the robot. The following factorization is then utilized.

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}).\, p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \quad (1)$$

This factorization enables us to first estimate the trajectory of the robot, and then compute the map, given that trajectory. If $x_{1:t}$ and $z_{1:t}$ are known, then the map m can be found efficiently since $p(m|x_{1:t}, z_{1:t})$ can be calculated analytically.

The posterior $p(x_{1:t}|z_{1:t}, u_{1:t-1})$ can be estimated over several potential trajectories with the help of a particle filter. A sample of particles are first selected where each particle represents a potential trajectory of the robot (proposal). The maps associated with each of the particles are then computed using the observations, and the trajectory that corresponds to them.

The following four steps summarizes the Improved RBPF for Map Learning.

Sampling: The new pose of a particle i is first estimated with the help of the motion model (odometry), and the previous pose of that particle. This prediction is then corrected using a scan-matching algorithm which incorporates the most recent observations, and then evaluates how well the new particles fit the map obtained so far. If the scan-matching reports a success, then a new set of sampling points are selected in an interval around the pose returned by the scan-matcher. Based on these points, the mean and the covariance matrix of the proposal are computed. The new pose of particle i is estimated from the Gaussian approximation $N\left(\mu_t^{(i)}, \Sigma_t^{(i)}\right)$ of the improved proposal distribution. If the scan-matcher reports a failure, then the initial proposal distribution will be used to estimate the new poses.

Importance Weighting: For each particle, an individual importance weight $w_t^{(i)}$ is assigned based on the importance sampling principle. Here, π is the improved proposal distribution. $w_t^{(i)}$ measures how well a particle i drawn from the improved proposal distribution represents the actual target distribution.

$$w_t^{(i)} = \frac{p\left(x_{1:t}^{(i)} \middle| z_{1:t}, u_{1:t-1}\right)}{\pi\left(x_{1:t}^{(i)} \middle| z_{1:t}, u_{1:t-1}\right)} \tag{2}$$

Adaptive Resampling: Resampling replaces particles of low importance rate with samples with high weight. An effective sample size N_{eff} is used to estimate how well the current particle set approximates the target posterior. N_{eff} is a measure of the dispersion of important weights and can be formulated as shown in Eq. 3. Here, $\widetilde{w}^{(i)}$ represents the normalized weight of the particle i. Resampling is done every time N_{eff} drops below the threshold of $N/2$ where N is the number of particles.

$$N_{eff} = \frac{1}{\sum_{i=1}^{n}(\widetilde{w}^{(i)})^2} \tag{3}$$

Map Estimation: The map that corresponds to each particle is updated based on the robot poses and the observations obtained so far.

Experiment

The goal of this study was to evaluate how well the Gmapping SLAM algorithm performed under various lidar noises. The approach that was taken for this was to compare the actual positions of certain critical points in a Gazebo environment, to their positions on the map published by Gmapping.

First, a virtual environment was created on Gazebo [10], and a turtlebot waffle was placed inside it, as shown below in Fig. 1.

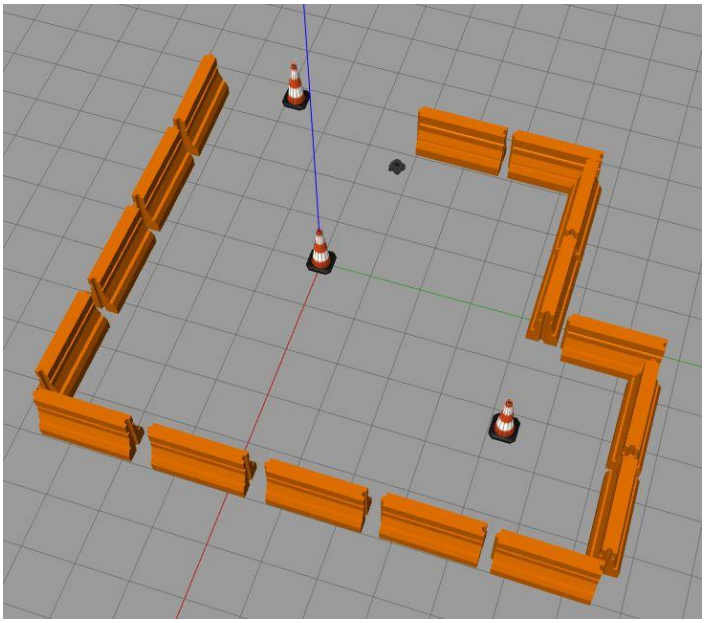

Fig.1 Gazebo environment with the turtlebot waffle inside it

The environment is an enclosed area because otherwise, the robot will get lost and will be unable to map the surroundings and localize itself effectively. The construction cones are assumed to be the critical points in the environment. To map the environment, the robot has to move based on control inputs that are given to it. For this, the turtlebot3_teleop [11] was used. The lidar noise was decided inside the turtlebot waffle URDF file, where the mean and standard deviations of the Gaussian lidar noise can be determined by the user. Gmapping was then used to perform SLAM, and rviz [12], was used to visualize the map that was being created by the robot. Two important topics that are published in Gmapping are map, and map_metadata. The topic map contains the message OccupancyGrid [13], which is a 2-D grid map, where each cell represents the probability of occupancy. The topic map_metadata contains the message MapMetaData [14], which has information about the characteristics of the OccupancyGrid such as the map resolution, map height, map width, map pose, and map origin. Two ROS subscribers were written; one where the OccupancyGrid was saved as an array inside a text file, and another, where MapMetaData was generated. The text file obtained from OccupancyGrid, and the information from MapMetaData was then used to plot the maps produced by Gmapping on MATLAB, and then find the positions of the critical points on the map. The errors between the actual positions of the critical points, and their positions inside the map were then compared for different lidar noises. In addition, the map generated on rivz was also saved using map_server [15].

Results

Initially, the lidar was assumed to have no noise, and the mapping was completed. The map obtained from map server (left), as well as the one generated using MATLAB (right) are shown below in Fig. 2. It has to be noted that the positions of the cones were assumed to be the positions of the centroids of the cluster of points corresponding to each of the cones.

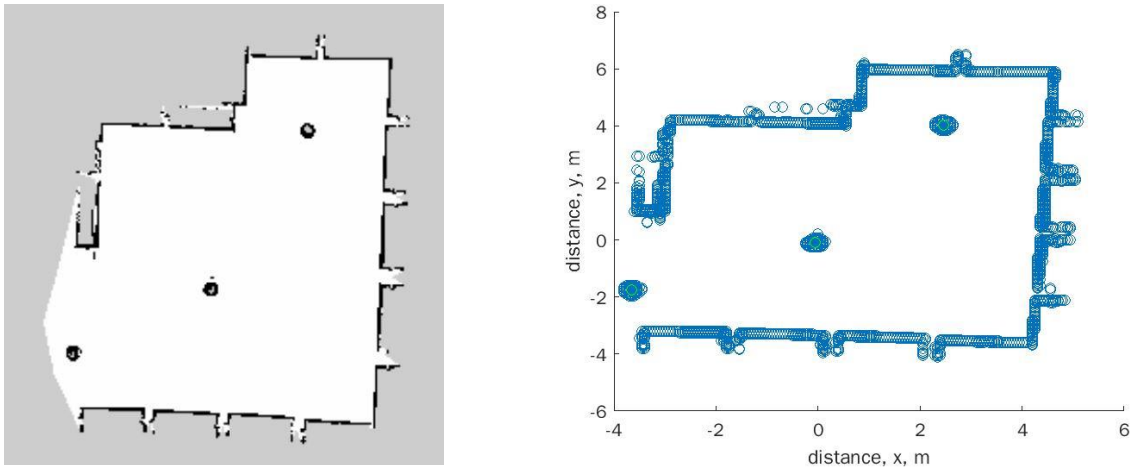

Fig. 2 Map generated by Gmapping when the gaussian lidar noise has standard deviation of 0 m

The gaussian lidar noise was then increased to have a standard deviation of 0.01 m. The map obtained from map server (left), as well as the one generated using MATLAB (right) are shown below in Fig. 3

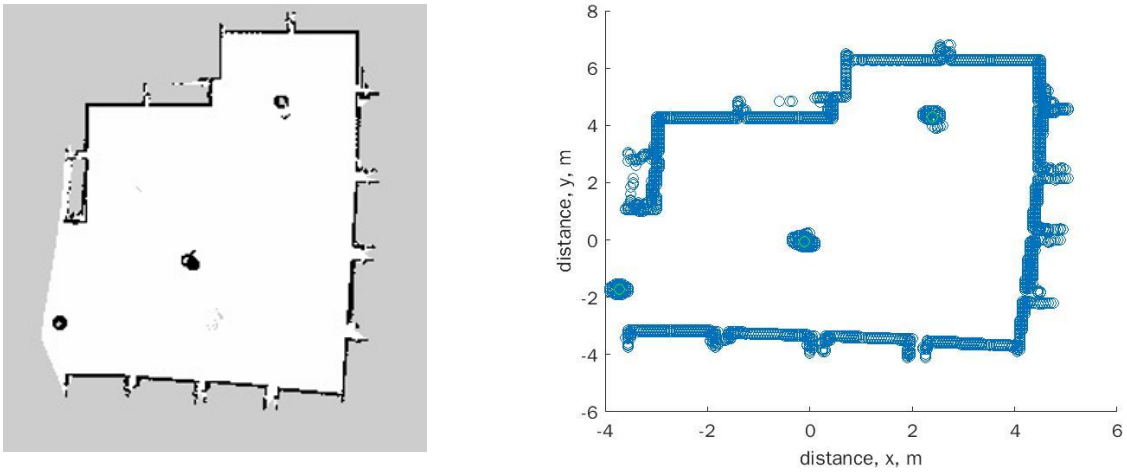

Fig. 3 Map generated by Gmapping when the gaussian lidar noise has standard deviation of 0.01 m

The gaussian lidar noise was increased further to have a standard deviation of 0.05 m. The map obtained from the map server (left), as well as the one generated using MATLAB (right) are shown below in Fig. 4

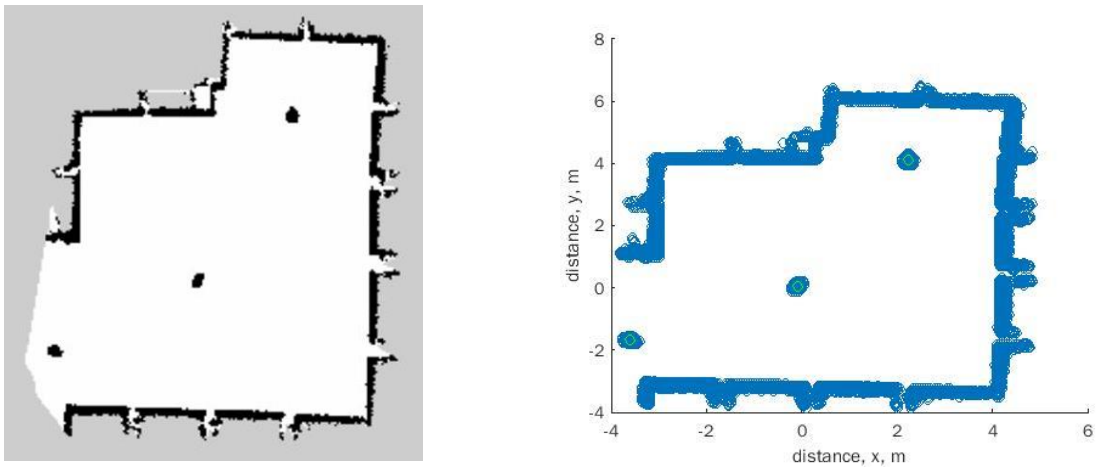

Fig. 4 Map generated by Gmapping when the gaussian lidar noise has a standard deviation of 0.05 m

The gaussian lidar noise was once again increased to have a standard deviation of 0.1 m. The map obtained from the map server (left), as well as the one generated using MATLAB (right) are shown below in Fig. 5

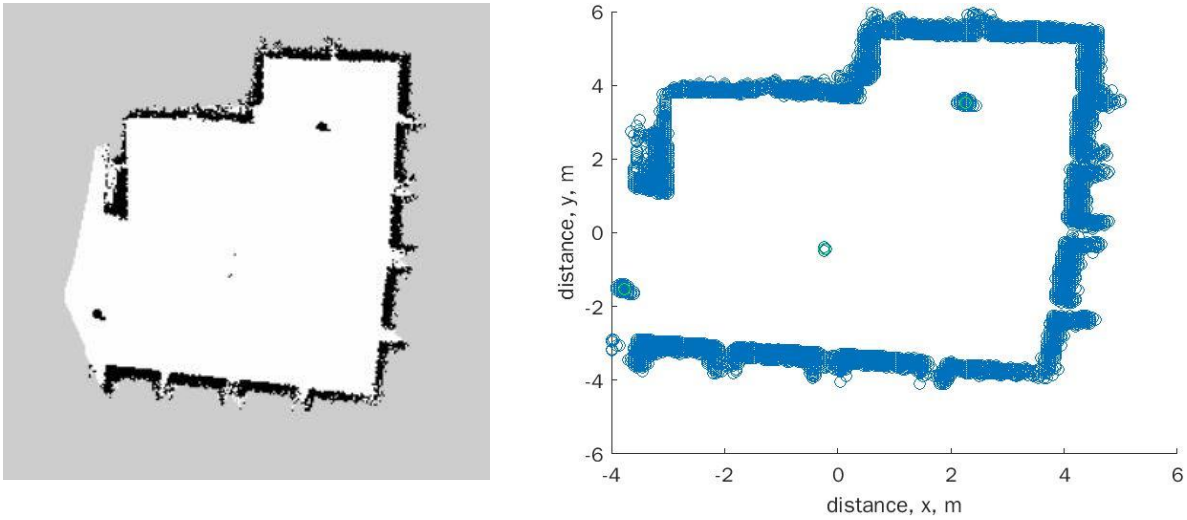

Fig. 5 Map generated by Gmapping when the gaussian lidar noise has a standard deviation of 0.1 m

Finally, the gaussian noise was increased one last time to have a standard deviation of 0.15 m. This map was not generated in MATLAB, as the critical points were not captured by the Gmapping algorithm. The below figure illustrates the map obtained from map server.

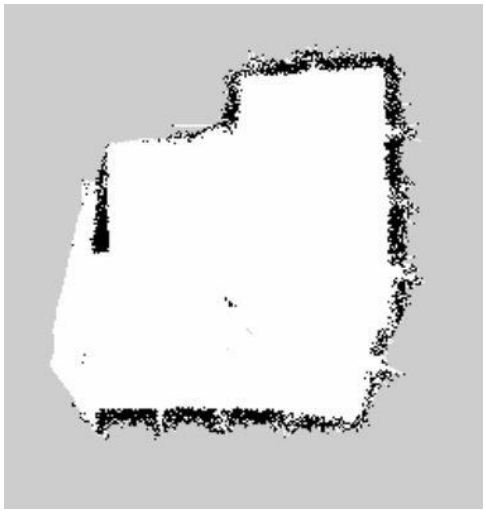

Fig. 6 Map generated by Gmapping when the gaussian lidar noise has a standard deviation of 0.15 m

The actual positions of the cones, as well their positions obtained by Gmapping, for lidar noises with standard deviations varying from 0 m to 0.1 m was plotted together as shown below in figure 7.

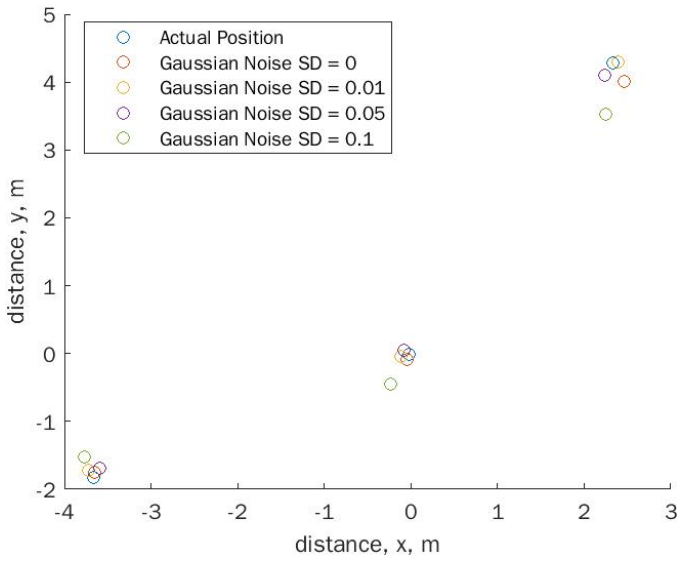

Fig. 7 Comparison of the positions of the cones for different Gaussian lidar noises

The errors in the positions of the cones for different Gaussian lidar noises are properly captured in the below table.

	Distance between actual position, and position on the map (m)		
Gaussian Noise SD (m)	Cone 1 (bottom left)	Cone 2 (middle)	Cone 3 (top right)
0	0.0683	0.0901	0.3005
0.01	0.1150	0.0962	0.0590
0.05	0.1531	0.0810	0.2020
0.1	0.3108	0.4896	0.7513

All the above results will now be explained in more detail.

The results obtained for no noise, as well as for a noise with standard deviation of 0.01 m were minimal, as one would expect, although for cone 3, the map corresponding to no lidar noise produced a much higher error than expected. This is possibly an error that was caused by insufficient measurements taken by the robot in the upper right region of the map. The mapping has to be performed again with no lidar noise, to make sure that the error for cone 3 would be much lower than what was obtained.

For the lidar noise with standard deviation of 0.05 m, the errors were higher than for a noise of 0.01 m. This is to be expected. Although the above table, as well as Fig. 7, provides the lowest error in the position of cone 2 for this lidar noise, it is not the most reliable result for finding the accuracy, due to the fact that the position of the cone was assumed to be the centroid of the cluster of cells corresponding to its occupancy. However, it may be the case that the actual boundaries of obstacles may have to be considered for certain tasks, in which case a better method has to developed to estimate the accuracy of the algorithm.

For the lidar noise with standard deviation of 0.1 m, the errors were the highest, as it should be. However, from the above table, it can be seen that the errors were significantly higher than before. This may have been because of the way the Gmapping algorithm works. The robot odometry is used to make a prediction of the pose of the robot, and the observation from the map is used to correct it. However, as the lidar noise becomes higher, it becomes harder to correct the prediction made from the odometry data and localize the robot. As a consequence, the map obtained will also contain higher errors. It must also be noted that the

number of points that were plotted for the occupancy of cone 2 were very low despite the multiple measurements taken by the robot during that trial. The insufficient number of cells is caused due to the high lidar noise which causes the lidar measurements of smaller objects such as the cones, to get spread across a larger area, making it very difficult for the robot to detect it. This effect can also be seen throughout the entire plot were the walls on the generated map is thicker, due to the widely spread distribution of the lidar measurements.

Conclusion

It was found that the Gmapping algorithm performed well, with errors mostly within the 0.05 m to 0.2 m range for a gaussian lidar noise with standard deviation varying between 0 m to 0.05 m. However, when the lidar noise standard deviation was increased to 0.1 m, it was found that the errors became more significant, ranging from approximately 0.3 m to 0.75 m. Overall, the results are what one would expect, with the map being more accurate with lower lidar noise. However, there were a few outliers such as the error of the position of cone 3 for no lidar noise, and the error of the position of cone 2 for lidar noise with a standard deviation of 0.05 m. It has to be noted that accurately determining the position of small objects on a map, is generally more challenging. However, Gmapping performed well enough to estimate their positions even under larger errors.

Future Work

There are a few things in this paper, that could be improved in future works. Firstly, the lidar noise was assumed to Gaussian. However, in reality, the lidar noise may have other forms of probability distributions. Hence, to truly test the Gmapping algorithm, the noise has to be non-Gaussian. Secondly, only the lidar noise was changed throughout the paper. However, the robot drift and the wheel encoder errors also play an important role in SLAM. Hence the combined effects of different lidar noises, as well as odometry errors must be taken into account. Finally, for a proper evaluation of Gmapping, real life experiments have to be conducted. The Gazebo simulations by itself, would not always provide realistic results.

References

[1] "gmapping - ROS Wiki." http://wiki.ros.org/gmapping (accessed Dec. 07, 2021).

[2] A. R. Khairuddin, M. S. Talib, and H. Haron, "Review on simultaneous localization and mapping (SLAM)," in *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Nov. 2015, pp. 85–90. doi: 10.1109/ICCSCE.2015.7482163.

[3] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun. 2001, doi: 10.1109/70.938381.

[4] D. Estler, "Path Planning and Optimization on SLAM-Based Maps," p. 55.

[5] C. M. Koehr, "EVALUATION OF ROS SLAM GMAPPING FOR EXTRATERRESTRIAL ROBOTIC MINING," p. 154.

[6] B. L. E. A. Balasuriya *et al.*, "Outdoor robot navigation using Gmapping based SLAM algorithm," in *2016 Moratuwa Engineering Research Conference (MERCon)*, Apr. 2016, pp. 403–408. doi: 10.1109/MERCon.2016.7480175.

[7] 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)*, Linkoping, Sweden, Oct. 2013, pp. 1–6. doi: 10.1109/SSRR.2013.6719348.

[8] Y. Abdelrasoul, A. B. S. H. Saman, and P. Sebastian, "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM," in *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, Sep. 2016, pp. 1–6. doi: 10.1109/ROMA.2016.7847825.

[9] G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters,” *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007, doi: 10.1109/TRO.2006.889486.
[10] “Gazebo.” http://gazebosim.org/ (accessed Dec. 07, 2021).
[11] “turtlebot3_teleop - ROS Wiki.” http://wiki.ros.org/turtlebot3_teleop (accessed Dec. 07, 2021).
[12] “rviz - ROS Wiki.” http://wiki.ros.org/rviz (accessed Dec. 07, 2021).
[13] “nav_msgs/OccupancyGrid Documentation.” http://docs.ros.org/en/api/nav_msgs/html/msg/OccupancyGrid.html (accessed Dec. 07, 2021).
[14] “nav_msgs/MapMetaData Documentation.” http://docs.ros.org/en/api/nav_msgs/html/msg/MapMetaData.html (accessed Dec. 07, 2021).
[15] “map_server - ROS Wiki.” http://wiki.ros.org/map_server (accessed Dec. 07, 2021).