www.dergipark.gov.tr
ISSN:2148-3736
El-Cezerî Fen ve Mühendislik Dergisi
Cilt: 8, No: 3, 2021 (1441-1454)

El-Cezerî Journal of Science and Engineering
Vol: 8, No: 3, 2021 (1441-1454)
DOI : 10.31202/ecjse.943364

ECJSE

Research Paper / Makale

Examining of the Effect of Geometric Objects on SLAM Performance Using ROS and Gazebo

Hamza AYDEMİR[a*], Mehmet TEKEREK[b], Mehmet GÖK[c]

[a]Informatics Systems Department, Kahramanmaraş Sütçüimam University, Kahramanmaraş, Turkey
[b]Informatics Systems Department, Kahramanmaraş Sütçüimam University, Kahramanmaraş, Turkey
[c]Distance Education Research and Application Center, Kahramanmaraş İstiklal University, Kahramanmaraş, Turkey
aydemirhamza1402@gmail.com

Received/Geliş: 26.05.2021 **Accepted/Kabul:** 19.07.2021

Abstract: An autonomous mobile robot needs a map of the environment and location information relative to the map. Simultaneous Localization and Mapping (SLAM) is a prediction process in which the autonomous mobile robot can use this map to determine its position while building a consistent map. The purpose of this study is to examine the effect of geometric objects on SLAM performance. In this direction, three different experimental areas including equilateral triangular prisms, square prisms and cylinders are designed in Gazebo. The fourth experiment area includes all three geometric objects used in the study. When the mapping times of the four experimental areas were compared, it was seen that the fastest scenario is achieved within triangular-only objects (9 min 55 sec) and the slowest within square (10 min 43 sec). In terms of measures, the generated map including the triangular prisms is the closest to the actual measures of the simulated area. Accordingly, the mapping error was calculated as 0.171 m^2 per 1 m^2 in an interior made of triangular prisms, and 0.682 m^2 in an interior made of square prisms. The obtained results show that the shapes of the geometric objects directly affect the performance of SLAM.

Keywords: Gazebo, Gmapping, ROS, SLAM

ROS ve Gazebo Kullanılarak Geometrik Cisimlerin SLAM Performansına Etkisinin İncelenmesi

Öz: Otonom bir mobil robotun gezinme için bir çevre haritasına ve haritaya göre konum bilgilerine ihtiyacı vardır. Eş Zamanlı Konum Belirleme ve Haritalama (SLAM), bir otonom mobil robotun, tutarlı bir harita oluştururken konumunu belirlemek için bu haritayı kullanabileceği bir tahmin sürecidir. Bu çalışmanın amacı, geometrik nesnelerin SLAM performansı üzerindeki etkisini incelemektir. Bu doğrultuda Gazebo ortamında eşkenar üçgen prizma, kare prizma ve silindir içeren üç farklı deney alanı tasarlanmıştır. Dördüncü deney alanı, çalışmada kullanılan üç geometrik nesnenin tümünü içermektedir. Dört deney alanının haritalama süreleri karşılaştırıldığında, en hızlı üçgen prizma (9 dakika 55 saniye) ve en yavaş kare (10 dakika 43 saniye) deney alanının haritasının oluşturulduğu görülmüştür. Oluşturulan haritada yapılan ölçümlerde gerçek ölçülere en yakın haritanın üçgen harita olduğu görülmüştür. Buna göre, üçgen prizmadan oluşan bir iç mekânda 1 m^2'de haritalama hatası 0,171 m^2, kare prizmadan oluşan bir iç mekânda ise 0,682 m^2 olarak hesaplanmıştır. Elde edilen bulgular, cisimlerin geometrik şekillerinin SLAM performansını direkt olarak etkilediğini göstermektedir.

Anahtar Kelimeler: Gazebo, Gmapping, ROS, SLAM

How to cite this article
Aydemir, H., Tekerek, M., and Gök, M., *"Examining of the Effect of Geometric Objects on SLAM Performance Using ROS and Gazebo"* El-Cezerî Journal of Science and Engineering, 2021, 8 (3); 1441-1454.

Bu makaleye atıf yapmak için
Aydemir, H., Tekerek, M., and Gök, M., *"ROS ve Gazebo Kullanılarak Geometrik Cisimlerin SLAM Performansına Etkisinin İncelenmesi"* El-Cezerî Fen ve Mühendislik Dergisi 2021, 8 (3); 1441-1454.
ORCID ID: [a]0000-0002-2657-3195; [b]0000-0001-6112-3651; [c]0000-0003-1656-5770

1. Introduction

Autonomous mobile robots that can operate in unknown environments are being developed to expand the application and spread of robots in the context of research and industry [1]. In this direction, it must be ensured that the robot can navigate in an unknown environment by using only its sensors when placed in any unknown point [2]. An autonomous mobile robot needs the map of its environment and the location information that indicates its location according to the map, to accomplish a navigation task [3-5]. Thanks to the map and location information, the robot can determine the shortest route from the starting point to the target [3] and increase the probability of reaching the target by constantly updating its route while avoiding the obstacles it encounters on the way [4].

The interdependence of location and mapping increases the complexity of the problem and requires these two problems to be solved simultaneously [5]. Simultaneous Localization and Mapping (SLAM) is a prediction process in which the autonomous mobile robot can build a consistent map step by step while at the same time using that map to determine its location [2, 8, 9, 10]. Algorithms such as Gmapping, Hector SLAM, and Karto SLAM have been developed to solve the SLAM problem in robotic applications [6].

- Gmapping is an open-source distance measuring-based SLAM algorithm developed in 2007. It is the most widely used SLAM algorithm, considered to be the most powerful algorithm for location and mapping worldwide [7]. Indoor and outdoor location and mapping can be performed with the Gmapping algorithm.
- Hector SLAM uses the high sampling frequency of modern two-dimensional LIDAR. It has a significant mapping effect in a small indoor setting. Hector SLAM requires a high-precision laser-based range finder [8].
- Karto SLAM is a graphics optimization method that uses a highly optimized and non-iterative matrix. Karto SLAM algorithm can obtain maps by using a low amount of memory space in static and dynamic environments [9].

The steps of the Gmapping SLAM algorithm are listed below [10].

Step 1. Landmark Subtraction: It is to distinguish objects that are different from the environment. These objects depict the obstacles that must be observed and avoided by the robot.

Step 2. Data Association: Matching data from multiple sensors to make sure identical landmarks are defined. This is a re-observation process. Thus, false or meaningless landmarks can be eliminated.

Step 3. Situation Prediction: It is to estimate the local position of the robot on the map created after landmark extraction and data association.

Step 4. Status Update: It is the recursive repetition of movement and location determination to create the map in real-time.

Singandhupe and La [11] attributed the SLAM problem has not been fully solved despite the great effort in the field of SLAM in the last 30 years, as SLAM is a prediction problem in their compilation study. Accordingly, they associated a good SLAM solution with the uncertainty of the data from the sensors and the targeted performance level. In this study, SLAM performance is associated with the mapping time and its convergence to the real environment measurements.

SLAM algorithms can be operated with the Robot Operating System (ROS). ROS is an open-source meta-operating system licensed under the Berkeley Software Distribution (BSD) license that

enables the efficient development of robot systems [11]. ROS includes libraries designed for robot applications, packages developed for various robot platforms, SLAM algorithm packages, navigation packages, 3D robot simulation environments, high-performance physics engines, and sensor add-ons with the option of noise pollution of measurement data [12].

SLAM algorithms provided by ROS or independently developed ones can be simulated and tested in simulation environments. Cost and time-saving robot simulation environments are used in robotics research to quickly and effectively test new concepts, strategies, and algorithms [13]. Also, simulation environments reduce the development cycle and can be applied in a versatile way for different environments [12]. The ROS-based Gazebo robot simulation environment is a powerful tool for testing robotic applications [13]. Takaya et al. [12] pointed out that there is no difference in robot behavior compared to simulation in the experiments they performed using robots in the Gazebo environment and the real world. Gazebo environment can simulate robot applications effectively by supporting with their findings. In addition to all these, the Gazebo environment provides researchers with an effective tool, as it provides ease of reconfiguration and enables the support of hardware components such as different sensors [11]. Rviz visualization tool, which works in harmony with Gazebo, enables simulation data to be displayed [14].

The TurtleBot3 mobile robot platform, the basic technology of navigation, using SLAM algorithms for real-world tests and simulation environments, has been developed by ROBOTIS [15]. TurtleBot3 is an open-source mobile robot based on ROS; consisting a 2D distance sensor (LIDAR), a single-board computer (SBC), real-time controller board, sensors, and components that provide mobility [10]. TurtleBot3, whose core technology is SLAM, Navigation, and Manipulation, can run SLAM algorithms for mapping tasks [16].

In this study, it is aimed to reveal the effect of geometric shapes of objects on SLAM performance. Four different indoor environments designed for this purpose. Time and map information data were obtained by using Gmapping SLAM algorithm implemented by using Turtlebot3 mobile robot. The measurements made on the map information data were compared with the data in the simulation environment. The measurement results show that our study will contribute to the understanding of the change in SLAM performance according to the geometric shapes of the objects in the environment of the robot.

2. Theoretical Framework

2.1. Simultaneous Localization and Mapping (SLAM)

Gmapping is a SLAM algorithm based on the Rao-Blackwellized particle filter that carries a separate map of the space of each particle. With the help of this algorithm, the robot creates a map of the space by sampling the particles with its sensor (LIDAR in the context of the study). The information on how far the robot has progressed in space is obtained by using the odometry sensor. SLAM problem can be expressed by using two factors: localization and mapping as given in Equation 1. Based on the Rao-Blackwellized particle filter for SLAM; $p(x_{1:t}, m|z_{1:t}, u_{0:t-1})$ is the posterior of robot's potential trajectories $x_{1:t}$ of the robot given its observations $z_{1:t}$ and robot's odometry measurements $u_{0:t}$, $p(m|x_{1:t}, z_{1:t})$ is the posterior over maps, and $p(x_{1:t}|z_{1:t}, u_{0:t-1})$ is the posterior over maps and trajectories [17].

$$p(x_{1:t}, m|z_{1:t}, u_{0:t-1}) = p(m|x_{1:t}, z_{1:t})p(x_{1:t}|z_{1:t}, u_{0:t-1}) \quad (1)$$

Gmapping creates a propagation pattern using probabilistic distribution methods based on the last observation site of the robot. In this way, a more accurate map is obtained by eliminating uncertain

situations. With the Rao-Blackwellized particle filter used in the method, resampling is performed at each update and the sample set representing the robot's trajectory and a map is updated. Sequentially; sampling, assignment based on the weight of importance; resampling, assigning according to the results, and estimating the map corresponding to each particle observation. As a result, low-weight particles are lost, and instead high-predictive particles are updated, increasing prediction accuracy.

With the method suggested by Doucet et al. [18], the resampling step was used in the Gmapping algorithm. According to this method, a selective stimulation technique is applied for the resampling process, taking into account the particles with significant weight value among the particles. The following equation Ne_{ff} expresses how well the particle weighs in the premise orbit. In Equation 2, $\widetilde{\omega}^{(i)}$ is the normalized weight of particle i. A resampling step is performed whenever Ne_{ff}'s particle count falls below half the number *N/2*. Thus, a more accurate map is provided while the risk of particle depletion is reduced.

$$Ne_{ff} = \frac{1}{\sum_{i=1}^{N}(\widetilde{\omega}^{(i)})^2} \tag{2}$$

The *turtlebot3_slam* package is used to run the gmapping algorithm with ROS on TurtleBot3. The diagram of this package is given in Figure 1. Accordingly, Gmapping package; subscribes to two nodes named *tf* and *scan*. *Scan* node refers to the data from the sensor. *Tf* node refers to the robot's motion mechanism (autonomous navigation in the context of the study). The data from these two nodes are calculated with the Gmapping algorithm and a topic called map is published. Map information can be saved by reading this topic (map) with the *map_server* package.

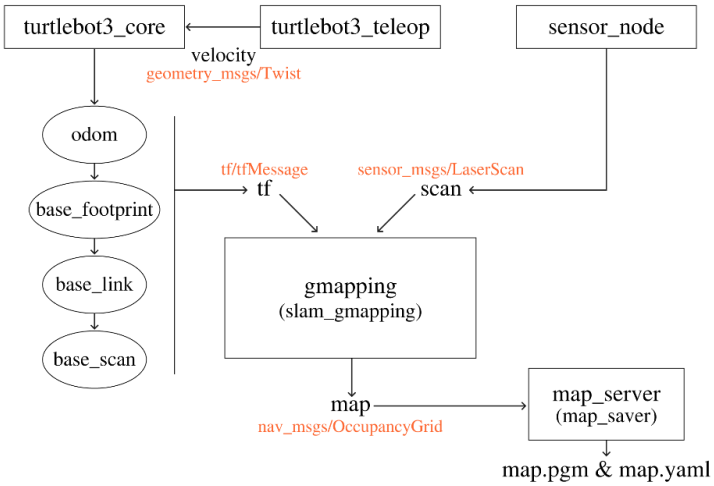


Figure 1. Schematic of the *turtlebot3_slam* package [19]

2.2. Exploration

For the robot to create the map of the environment, it must navigate through the environment. The navigation process can be achieved by making random displacement movements or by being controlled by an operator, or it can be performed autonomously. ROS package named *explore_lite* is used for automatic map creation within this study. Ease of configuration and low resource consumption has been effective in choosing this package [20]. With this package, the robot makes greedy frontier-based exploration. Greedy frontier-based exploration means the robot will continue to explore its surroundings until there are no frontiers left. Figure 2 shows the diagram of the

explore_lite package. Accordingly, the package subscribes to nav_msgs/OccupancyGrid and map_msgs/OccupancyGridUpdate messages issued by the Gmapping algorithm.

In this way, the unexplored regions on the map are determined by following the map created by Gmapping. The package can use the map published by move_base (<move_base>/global_costmap/costmap) or the map created with SLAM. Since the maps created with SLAM contain less noise, the map created with the SLAM algorithm (Gmapping) was used for the *explore_lite* package within the scope of the study.

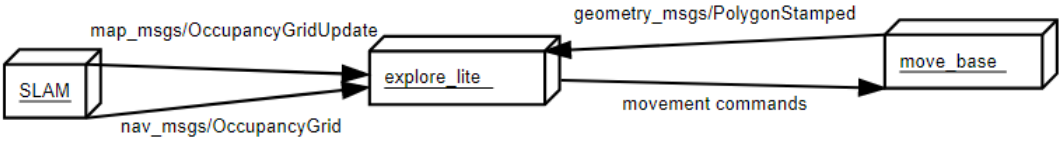


Figure 2. Diagram of the *explore_lite* package

Autonomous navigation task assigns frontiers to the boundaries between unexplored and explored areas. The robot starts to navigate unknown areas of the map by moving to the frontiers one by one until there are no frontiers remaining. The fact that there is no frontier means that the robot finished exploring the entire environment. In this case, the mapping process is completed and the constructed map is saved.

2.3. Map

The map created with the Gmapping SLAM algorithm is saved using the *map_server* package. Saving the map creates two files with the extension .yaml and .pgm containing the map information. YAML (YAML Ain't Markup Language) is a human-friendly, cross language and Unicode based data serialization language [21]. According to this; the YAML file contains map data in text format, while the PGM (Portable Gray Map) file contains pixel information in image format. YAML file contains variables named image, resolution, origin, occupied_thresh, free_thresh, and negate. The scale of the map in picture format of the indoor is kept in the variable named resolution in this file. According to this; resolution takes a default value of 0.5. This value means that the length of each 1 pixel in the image file is 5 cm [22]. To calculate the area of the indoor, it is derived that 1 pixel (5cm x 5cm) in the map image is 25 cm^2, based on the square form of pixels.

2.4. Robot

Building a map using the Gmapping algorithm requires an appropriate robot and a place where the robot can navigate.

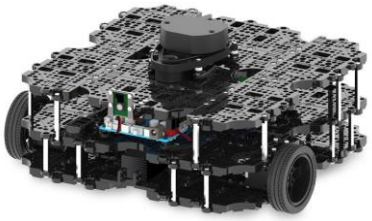


Figure 3. TurtleBot3 Waffle Pi robot

TurtleBot3 Waffle Pi, an open-source mobile robot based on ROS that can be used for educational and various research purposes, was used in this study. The image of the robot is given in Figure 3.

In the experimental tests, the Turtlebot3 Waffle Pi model is opted because it is well documented and provides sufficient technical structure for SLAM and navigation applications in an economical way. The technical features of the robot are given in Table 1.

Table 1. Technical features of the TurtleBot3 Waffle Pi robot [16]

Features	Explanation
Maximum translational velocity:	0.26 m/s
Maximum rotational velocity:	1.82 rad/s (104.27 deg/s)
Size (L x W x H):	281mm x 306mm x 141mm
Weight (SBC + Battery + Sensors)	1.8kg
Expected operating time:	2h
Expected charging time:	2h 30m
SBC (Single Board Computers)	Raspberry Pi 3 Model B and B+
Actuator:	Dynamixel XM430-W210
LDS (Laser Distance Sensor):	360 Laser Distance Sensor LDS-01
MCU (Mounted on Controller Board):	32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
IMU (Inertial Measurement Unit):	Gyroscope 3 Axis Accelerometer 3 Axis Magnetometer 3 Axis

2.5. Navigation Stack

When considered at the conceptual level, the navigation stack receives information from odometry and sensor nodes and sends speed commands to the actuator to activate the robot [23]. In Figure 4, the diagram of the navigation stack used for the realization of the robot's movement is given. The white components in the diagram are mandatory components, gray components are optional components, and blue components must be created for each robot platform.

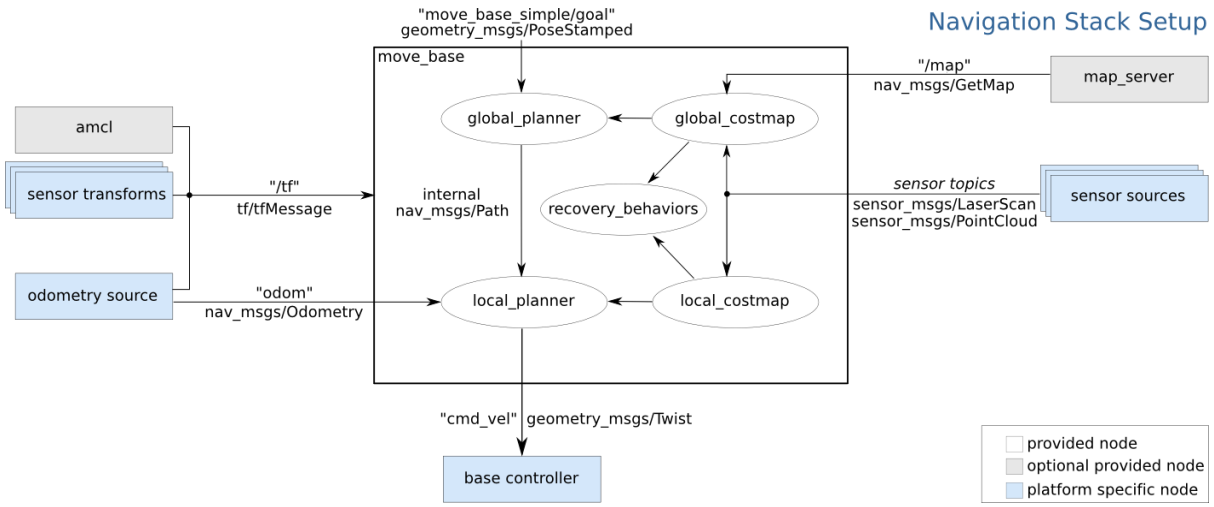


Figure 4. ROS Navigation Stack diagram

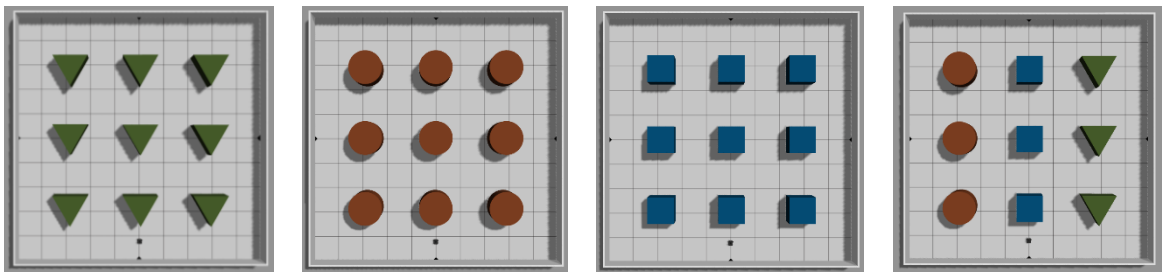
For the navigation stack, information about the relationships between coordinate points using the robot's *tf* is published. In addition, information from the sensors is used to avoid obstacles in the environment. For this, sensor_msgs/LaserScan or sensor_msgs/PointCloud messages are published through ROS. Odometry information is publish using *tf* and nav_msgs/Odometry message. Velocity commands are sent to the actuators that make the robot move, using the geometry_msgs/Twist message in cmd_vel. Finally, for the navigation stack, map information is not required, but as part of the study, the navigation stack is run using the map of the indoor. Meanwhile, the robot can avoid obstacles via the Dynamic Window Approach (DWA). DWA is a popular method for

obstacle avoidance planning and obstacle avoidance [19]. As a prerequisite for using the navigation stack; the robot operating with ROS, must have a *tf* transformation tree and must publish sensor data. Gazebo robot simulation and TurtleBot3 Waffle Pi robot used in the research meet these requirements.

3. Navigation Stack

3.1. Creating a Simulation Environment in Gazebo

Within the scope of the study, four experimental areas were designed in the Gazebo robot simulation environment (Figure 5). Experiment 1, includes 9 equilateral triangular prisms (Figure 5a); Experiment 2, includes 9 cylinders (Figure 5b); Experiment 3, 9 square prisms (Figure 5c) and Experiment 4, includes 3 equilateral triangular prisms, 3 square prisms and 3 cylinders (Figure 5d). In all experimental areas, the perimeter of the area is surrounded by 14.5 meters long and 1 meter high walls in a square form in order for the robot to navigate the equal area during the simulation. In all experiments, 9 columns were placed in this area, which is 210.25 m^2. The geometric objects placed in the simulation environment are designed in metric units in the 3D solid modelling environment. According to this; an equilateral triangular prism with a side length of 2 meters, a square prism with a side length of 1.5 meters, and a cylinder with a radius of 0.955 meters were modelled. All geometric bodies have a circumference of 6 meters and a height of 1 meter. By keeping the circumferences of geometric objects equal, it is aimed to prevent the change of the area scanned by the sensor to affect the mapping error. Designs were saved in .dae format supported by the Gazebo. Before starting the experiment, length measurements of geometric objects were made in Gazebo simulation and Rviz visualization environment and the desired measurements were confirmed. In Figure 5, the top view of the four experimental fields is given. The robot is positioned in the simulation environment where the experiment is carried out, at the same point in all experiments with the same angular feature.



(a) (b) (c) (d)

Figure 5. Top view of the experimental areas created in Gazebo robot simulation environment. (a) Equilateral triangular prism experimental area (Experiment 1), (b) cylinder experimental area (Experiment 2), (c) square prism experimental area (Experiment 3) and (d) mixed experimental area (Experiment 4)

3.2. Exporting Robot to Gazebo

The Gmapping SLAM algorithm used in the study was run on a robot named TurtleBot3 Waffle Pi. Within the scope of the study, the robot was transferred to the experimental areas designed in the Gazebo robot simulation environment via the *slam_research* package. With this process, the position and angle of the robot on all maps were ensured to be equal. No changes were made to the robot and to the experimental area. The model of the robot of the Rviz visualization tool is given in Figure 6.

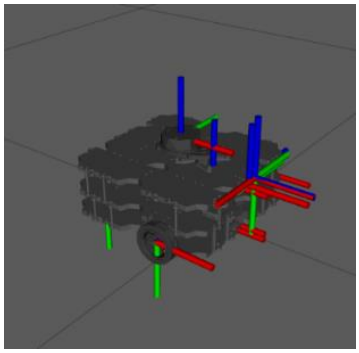


Figure 6. TurtleBot3 Waffle Pi in Rviz visualization tool

3.3. Performing the Simulation

The *slam_research* ROS package was created within the scope of the study. The file structure of the package is shown in Figure 7. The package contains 1 launch file and 4 world files. Accordingly, the launch file runs by taking four different world files as parameters. Each world file corresponds to a simulation environment prepared in the Gazebo. It is called from the *turtlebot_gazebo* package to the robot simulation environment. By the following parameters received by the robot constant, it was aimed to prevent the experiment from being affected by mentioned variables.

Figure 7. The file structure of the *slam_research* package

The parameters in the file named gmapping_params.yaml have been changed so that the sensor noises in the Gazebo robot simulation environment do not affect the study result. Accordingly, the minimumScore, srr (linear noise component – *x* and *y*), srt (linear to angular noise component), str (angular to linear noise component), and stt (angular noise component - *theta*) parameters in the file containing 29 parameters were updated. The default value and updated value of these parameters are shared in Table 2. The minimumScore parameter is used to evaluate the result of the scan match. The minimumScore parameter can prevent *jumping* exposure estimates in wide open areas when using laser scanners with limited range [24]. The odometry model error is represented with four parameters srr, srt, str, and stt which defines the variance of the zero-mean Gaussian distribution from which the odometry model error is sampled [25]. In summary; specifies the rotation error due to srr rotation, the translation error due to srt rotation, the rotation error due to str translation, and the translation error due to stt translation.

Table 2. Parameter change in *gmapping_params.yaml* file

Parameters	Default values	Updated values
minimumScore:	50	5000
srr:	0.1	0.00001
srt:	0.2	0.00002
str:	0.1	0.00001
stt:	0.2	0.00002

The map information view before and after the change in the gmapping configuration is given in Figure 8. As depicted in the figures, the noise parameters that mimic real-world sensor noises were changed to create a more standard map.

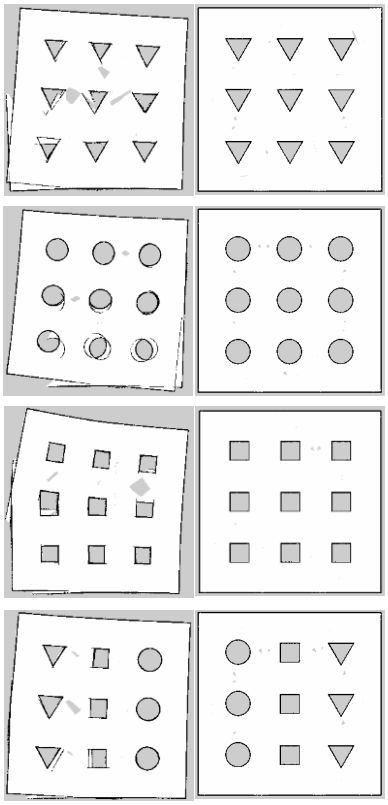


Figure 8. The map information view before and after the change in the *gmapping* configuration

The experimental process was followed by the researchers on two separate screens. To examine the findings in detail, the whole process was recorded as video file by using screen capture. In all maps, when the robot mapping process is completed, that is, when there are frontiers to explore, the map information has been saved and shared in the findings section. Using the map information obtained at the end of the experiment, measurements were made with a Python script file prepared by the researchers. In the measurements phase, the measures of the environment were calculated by reading the pixel information on the map. The data obtained from the measurements are given in the results section. No intervention was made to the environment or robot during the experiment.

4. Findings

At the end of the experiment, approximately 50 minutes of screen capture video was recorded and examined by the researchers. The compiled version of these videos can be accessed at the following link: https://youtu.be/cgWJMFwMlF0. During the review, screenshots of the map being created was taken at the 1st, 3rd, and 5th minutes in the Rviz visualization tool. This task was repeated for each experiment. Image outputs of the Gmapping SLAM algorithm in Experiment 1, 2, 3 and 4 are given in Figure 9, 10, 11 and 12, respectively. At each figure map creation process and the completed map are depicted, from left to right, respectively. The map information has been recorded when the robot is finished navigating. According to this; In the first three images in the figures, the gray areas represent the explored area, the blue lines represent the borders of the unexplored areas, and the green dots represent the frontiers. In the last map information image, the white area is the explored area, the black lines are the boundaries of the obstacles, and the gray areas are the unexplored areas.

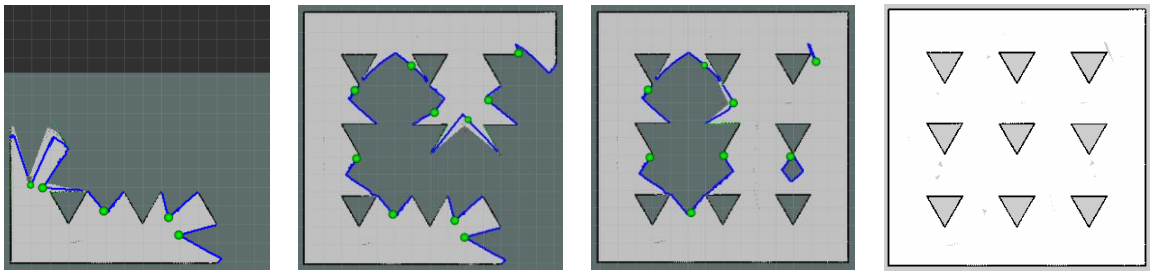


Figure 9. Screenshot of the map creation process and map output in Experiment 1

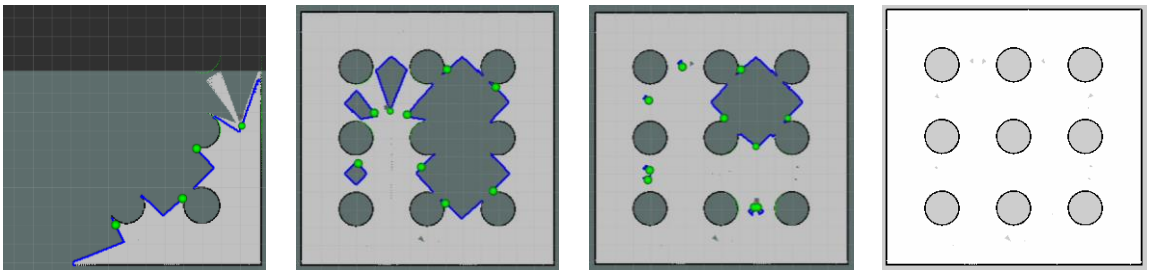


Figure 10. Screenshot of the map creation process and map output in Experiment 2

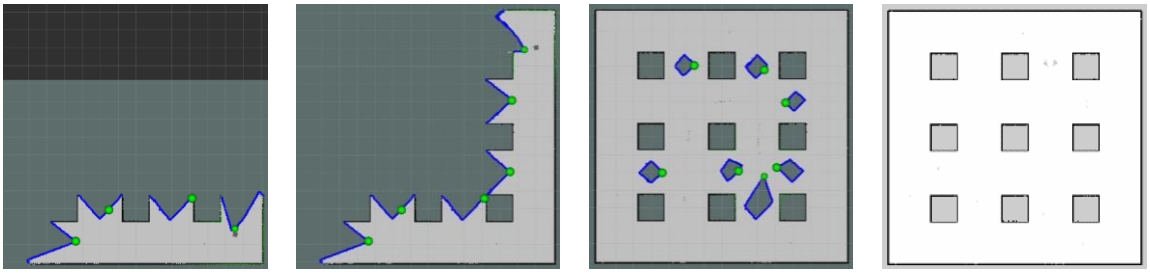


Figure 11. Screenshot of the map creation process and map output in Experiment 3

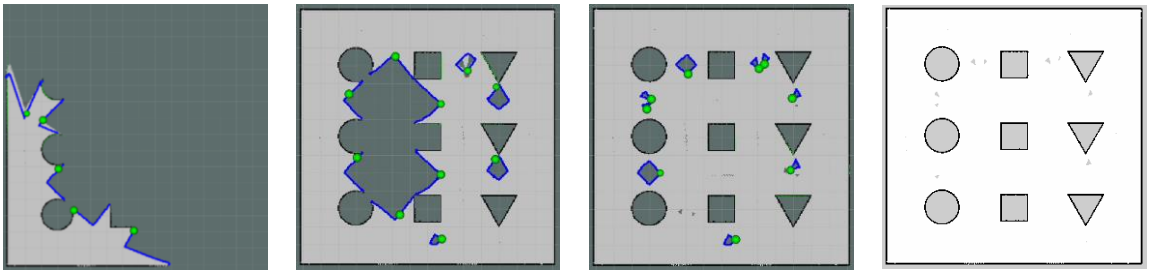
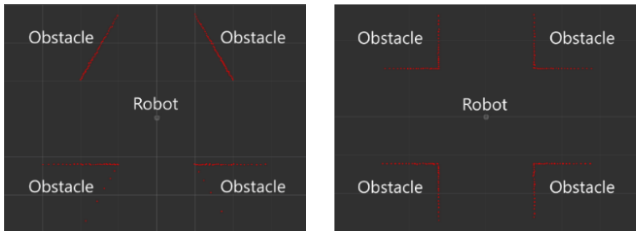


Figure 12. Screenshot of the map creation process and map output in Experiment 4

The elapsed time for the robot to create the perimeter of the experimental area and the 9 columns in the experimental area for each map using the Gmapping SLAM algorithm is given in Table 3.



(a) Experiment 1 (b) Experiment 3

Figure 13. Rviz view of the LIDAR beam blocked by geometric shapes

Each experiment is repeated for four times and mean of elapsed times are given in this table. We observe that the map creation time for each experiment continuously increases due to the amount of LIDAR beam blocked by geometric shapes accordingly. As can be seen in Figure 13, the LIDAR trace in Experiment 1 (Figure 13a) can scan more areas than in Experiment 3 (Figure 13b) at the same location due to the geometric shape of the obstacles.

Table 3. Experiments and mapping times

Experiment No	Elapsed Time
Experiment 1	09 min 55 sec
Experiment 2	10 min 07 sec
Experiment 3	10 min 43 sec
Experiment 4	10 min 04 sec

By comparing the size information in the Gazebo robot simulation environment with the map information obtained as a result of the experiment, the degree of overlap of the map with the real environment can be examined. In this direction, the measurements of the geometric objects were made in meters in the Gazebo simulation environment, which represents the real world.

Table 4. Gazebo robot simulation environment and map comparison of the measurement information of experiments

	Actual size of obstacle area	Calculated size of obstacle area	Mapping error (%)
Experiment 1	36 m^2	29.84 m^2	17,11
Experiment 2	25.2 m^2	39.88 m^2	-58,25
Experiment 3	20.25 m^2	34.08 m^2	-68,30
Experiment 4	27.15 m^2	34.62 m^2	-27,51

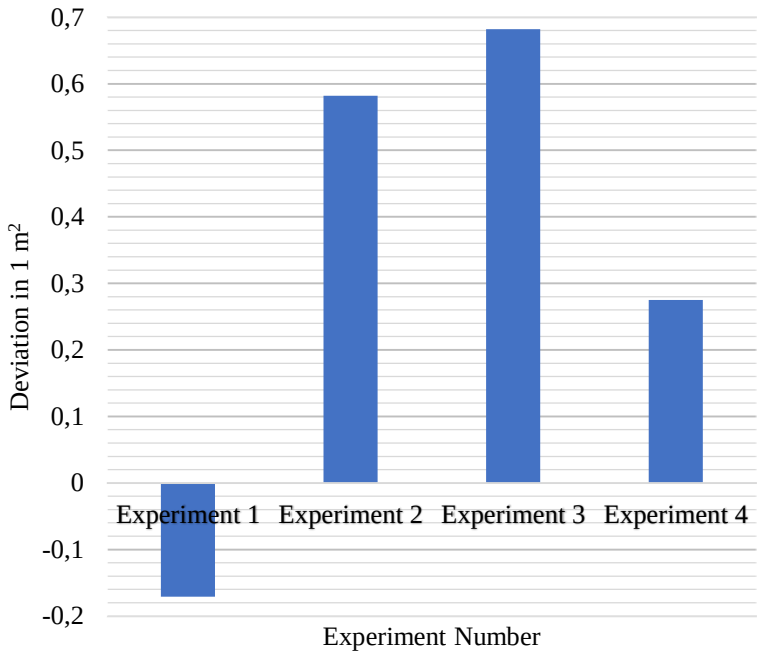


Figure 14. Mapping error (Deviation from actual sizes) in 1 m^2.

The measurements of the geometric objects in the map, which was built step by step in the Rviz simulation environment using the Gmapping SLAM algorithm, were measured in meters. The data

obtained are given in Table 4. The area values in Table 4 are calculated as the mean values of four repetitions for each experiment. Here we use the amount of area covered by obstacles as a criterion to evaluate mapping error. The area covered by the obstacles was calculated by counting the pixels of the obstacles on the image file.

Considering the data in Table 4, it can be calculated how far from the real values in 1 m^2 for each map type. Accordingly, the deviation in 1 m^2 is visualized in the graphic Figure 14. When the data are interpreted; Mapping error in an indoor consisting of triangular columns is 0.171 m^2 in 1 m^2, mapping error in an indoor consisting of square columns is 0.682 m^2 in 1 m^2, in an indoor consisting of cylinder columns, mapping error is 0.582 m^2 in 1 m^2 and an indoor consisting of mixed columns. Indoor mapping error deviated by 0.275 m^2 in 1 m^2. The size of inner columns in an indoor that consists of only triangular columns is less calculated than the actual measurements.

5. Conclusion

In this study, which was carried out to examine the effect of geometric objects on SLAM performance, four indoor environments were designed in the Gazebo robot simulation environment. In these enviroments; an equilateral triangular prism, square prism, and cylinder objects are used. In the experimental areas where these 3 geometric objects are located separately, TurtleBot3 Waffle Pi created 3 maps using the autonomous mobile robot Gmapping SLAM algorithm. The same operation was carried out in a mixed experiment area where 3 geometric objects were located together and as a result of this process, 1 map was obtained. The experimental process was recorded and the collected data are given under the heading of findings.

When the time to create the map of the four experimental areas is compared, it is seen that the fastest map was obtained from the triangular test area. The map of this area was created in 9 minutes and 55 seconds. When the maps are examined in terms of creation speed, mixed map, cylinder, and square map, respectively.

Measurements made by taking into account the map information of the geometric objects in the Gazebo and the map were compared. As a result, the triangular map's convergence performance to the real values was higher than the other maps. After the triangular map, the map with the highest performance is the hybrid map. Then comes the cylinder and square map. The mapping error is calculated as 0.171 m^2 in 1 m^2 in an indoor environment consisting of triangular columns, and 0.682 m^2 in 1 m^2 in an indoor consisting of square columns.
As a result of the study, it is seen that the map that is the fastest and closest to the real values was created in the indoor environment consisting of triangular columns. The rendering time and error rate of the cylinder and square maps are higher.

In our further studies, real-world experiments will behold by using those primitive shapes in a comparative manner related to simulations. In addition to this, path planning optimizations according to different maps containing mixed types of geometric shapes will be studied.

Authors' Contributions

All authors contributed equally to this study. All authors read and approved the final manuscript.

Competing Interests

The authors declare that they have no competing interests.

References

[1]. Elfes, A., Using occupancy grids for mobile robot perception and navigation, Computer, 1989, 22 (6), 46-57.

[2]. Castellanos, J. A., Neira J. and Tard´os J. D., Limits to the consistency of EKF-based SLAM, 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, (2004).

[3]. Brooks, R. A., A robust layered control system for a mobile robot, IEEE Journal of Robotics and Automation, 1986, 2 (1), 14-23.

[4]. Crowley, J. L., Navigation for an intelligent mobile robot, IEEE Journal of Robotics and Automation, 1985, 1 (1), 31-41.

[5]. Saeedi, S., Trentini, M., Seto, M. and Li, H., Space Robotics, Part II Editorial, Journal of Field Robotics, 2007, 24 (4), 273-274.

[6]. Xuexi, Z., Guokun, L., Genping, F., Dongliang, X. and Shiliu, L., SLAM algorithm analysis of mobile robot based on lidar, 38th Chinese Control Conference (CCC), Guangzhou, China, (2019).

[7]. Santos, J. M., Portugal, D. and Rocha, R. P., An evaluation of 2D SLAM techniques available in robot operating system, IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Linkoping, Sweden, (2013).

[8]. Jiang, G., Yin, L., Jin, S., Tian, C., Ma, X. and Ou, Y., A simultaneous localization and mapping (SLAM) framework for 2.5D map building based on low-cost LiDAR and vision fusion, Applied Sciences, 2019, 9 (10), 2105.

[9]. Gao, Q., Jia, H., Liu, Y. and Tian, X., Design of mobile robot based on cartographer SLAM algorithm, 2nd International Conference on Informatics, Control and Automation (ICA 2019), Phuket Island, Thailand, (2019).

[10]. Singh, D., Trivedi, E., Sharma, Y. and Niranjan, V., TurtleBot: Design and hardware component selection, International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, (2018).

[11]. Zhang, M., Qin, H., Lan, M., Lin, J., Wang, S., Liu, K., Lin, F. and Chen, B. M., A high fidelity simulator for a quadrotor UAV using ROS and Gazebo, 41st Annual Conference of the IEEE Industrial Electronics Society (IECON), Yokohama, Japan, (2015).

[12]. Takaya, K. Asai, T., Kroumov, V. and Smarandache, F., Simulation environment for mobile robots testing using ROS and Gazebo, 20th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, (2016).

[13]. Qian, W., Xia, Z., Xiong, J., Gan, Y., Guo, Y., Weng, S., Deng, H., Hu, Y. and Zhang, J., Manipulation task simulation using ROS and Gazebo, IEEE International Conference on Robotics and Biomimetics (ROBIO), Bali, (2015).

[14]. Karalekas, G., Vologiannidis, S. and Kalomiros, J., Europa: A case study for teaching sensors, data acquisition and robotics via a ROS-based educational robot, Sensors, 2020, 20 (9).

[15]. Ackerman, E., IEEE Spectrum, https://spectrum.ieee.org/automaton/robotics/diy/interview-turtlebot-inventors-tell-us-everything-about-the-robot (accessed 26.05.2021).

[16]. ROBOTIS, ROBOTIS e-Manuel, http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/ (accessed 26.05., 2021).

[17]. Stachniss, C., Grisetti, G. and Burgard, W., Information Gain-based exploration using Rao-Blackwellized particle filters, Robotics: Science and Systems, 2005, 2, 65-72.

[18]. Doucet, A., Freitas, N. d. and Gordon, N., An introduction to sequential Monte Carlo methods, New York: Springer, 2001, 3-14.

[19]. Pyo, Y., Cho, H., Jung, R. and Lim, T., ROS robot programming, Seoul: Robotis, 2017.

[20]. Hörner, J., Map-merging for multi-robot system, Barchelor Thesis, Computer Science, Charles University, 2016.

[21]. Ben-Kiki, O., Evans, C. and Net, I. d., YAML Ain't Markup Language (YAML™) Version 1.2, https://yaml.org/about.html, (accessed 25.11., 2020).
[22]. ROS Wiki, Map_server Package, http://wiki.ros.org/map_server, (accessed 26.05., 2021).
[23]. ROS Wiki, Navigation Stack, http://wiki.ros.org/navigation/Tutorials/RobotSetup, (accessed 26.05., 2021).
[24]. Gerkey, B., slam_gmapping, http://docs.ros.org/en/hydro/api/gmapping/html/index.html (accessed 26.05., 2021).
[25]. Popovic, G., Orsulic, J., Miklic, D. and Bogdan, S., Rao-Blackwellized particle filter SLAM with prior map: an experimental evaluation, ROBOT 2017: Third Iberian Robotics Conference, Seville, Spain, (2018).