ISSN 2278-3091

Volume 9, No.1.3, 2020

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse6091.32020.pdf

https://doi.org/10.30534/ijatcse/2020/6091.32020

Exploring Mapping and Avoidance Simulation for Mobile Robot

Mohammad Ikmal Shamin Abdul Rahman[1], **Sofianita Mutalib**[2], **Mohamad Amin Mohamad Zambri**[3], **Shuzlina Abdul-Rahman**[4]

[1]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia, mdikmal96@gmail.com

[2]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia, sofi@fskm.uitm.edu.my

[3]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia

[4]Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia, shuzlina@fskm.uitm.edu.my

ABSTRACT

The use of mobile robot simulation is widely seen in various purpose. The navigation of the mobile robot is important to make the simulation process success. Simultaneous Localization and Mapping (SLAM) is one of the important features in a mobile robot navigation to enable an autonomous robot to map its surrounding and localize itself in real time. This paper presents a simulation study that investigates the algorithms that could be used in localization and mapping processes based on Karto and GMapping SLAM algorithms. The algorithms were tested on Robot Operating System (ROS) simulation in Linux environment. Experimental results from the Karto SLAM algorithm showed that the map produced is more precise and reliable for localization compared to its counterpart. In addition, the notification of distance is provided to give information on obstacle. Our results showed that Karto SLAM algorithm produced a better position estimation and can determine the direction of where the mobile robot is moving autonomously without colliding in any collision. Further studies on other SLAM algorithms could be done to generate more accurate results to localize the autonomous vehicle.

Key words : Karto, Mobile Robot, Obstacle Avoidance, SLAM.

1. INTRODUCTION

Mobile robot is a technology that would be very useful to undertake many tasks generally carried out by humans. Unlike humans, robots do not have (the five) senses like humans do such as eyes to see, ears to hear, nose to smell, tongue to taste and hands to touch objects [1]. Human beings use their senses to perceive their environment and acquire information through them. On the other hand, mobile robots depend on sensing devices to obtain information on unknown environment [1], [2]. There are many types of sensing devices that could be implemented on a mobile robot so that it can sense and scan a wider range of the environment. These sensors include odometers, Global Position System (GPS), cameras, Light Detection and Ranging (LiDAR), Sound Navigation and Ranging (Sonar), Laser range finders and Inertia Measurement Units (IMU) [3],[4].

In order to enable a robot moves by itself to perform its tasks, a map of the environment is the most basic need for the robot [5]. Unfortunately, knowing the map of the environment alone is not enough. The robot should also be able to 'know' its location in its environment. Thus, Simultaneous Localization and Mapping (SLAM) is used to build the map whilst at the same time locate the position of a mobile robot at an unknown position. One of operators for a mobile robot is known as the Robot Operating System (ROS). Mobile robot is used to build the map of a new environment, locate itself in the map built and do some navigation towards a given point. ROS is usually used to create robot applications. A mobile robot is a compatible robot to demonstrate the functionality of ROS. Code sharing could be used in ROS and it also supports an open-source robotics nature. This could help in improving the robotic industry as there are many codes being improved when other researchers extend the results of their research. To find the right software to integrate in a robot system is easier when using ROS. By using mobile robot simulation in ROS, lots of test could be done such as driving the robot around the environment, build the map and visualize it in e [5],[6]. Many tasks could be carried out using mobile robot for testing the selected algorithms so that they could be implemented in a new design or process. Besides that, simulation is very useful to simplify the process of learning algorithms particularly when collecting and annotating large volumes of real data are both impractical and expensive. The simulation domain can be procedurally constructed to specification, allowing tests to be conducted especially under the said (impractical and expensive) conditions.

/.

In order to navigate autonomously, an autonomous vehicle need to know the precise location of its current state. This is what is meant by localization or likewise known as local state, which refers to the current location of an autonomous vehicle as it navigates to the other places. In short, to require a precise location, a localization process has to be within an a priori map [3],[4]. Wang et al. [3] explained in their article that metadata is embedded into an a priori map, which transforms the complicated perception task into localization problem rather than using a vehicle's sensor to detect lane, markings and traffic signs clearly. In general, SLAM processes mainly consist of two steps. The first step is estimating the position of the autonomous vehicle at k- moment based on the position of the autonomous vehicle at k-1 moment and the motion model. K-moment refers to the current moment, while k-1 moment means to the previous moment. The second step is, at k-moment, by using the estimated position, the observed model is used to estimate the environment landmarks. Put it differently, this process is also known as dead reckoning, which refers to the process of estimating the current location based on the previous location [4],[5].

This paper presents a case study based on a SLAM algorithm in a simulated environment using Turtlebot in ROS package. The SLAM algorithm tested is the Karto SLAM that is a graph variant algorithm and GMapping. The next section will elaborate on the SLAM algorithms and, later in Section 3 is the Methodology for the simulation experiments. Section 4 presents the Results and Discussion, and finally the Conclusion is found in Section 5.

2. RELATED STUDIES ABOUT SLAM

There are many algorithms that can be used in SLAM. Two main branches of SLAM algorithms are Filter-based and Optimization-based-SLAM. Figure 1 shows an overview of the SLAM algorithms. Some examples of SLAM methods that utilize the Filter-based approach are the Kalman Filter (KF), Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) [7], [8].

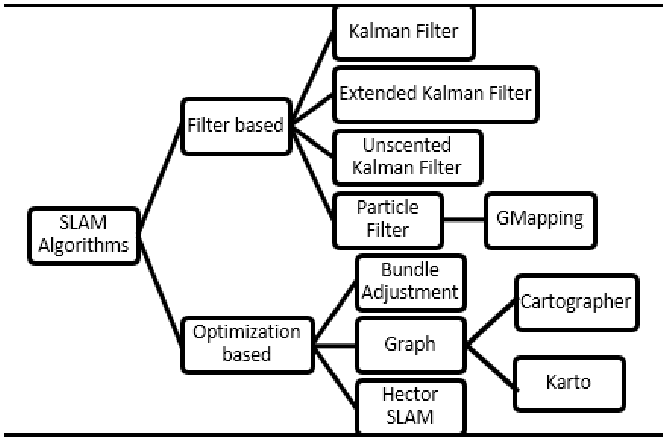


Figure 1: Overview of SLAM Algorithms

In the Filter-based category, the underlying principle is formed based on the Bayes' Theorem for the filter and it functions in a two-step process iteratively [8]. The first step is to utilize an evolution model and the control inputs to generate estimations of the vehicle's pose as well as the map states. The second step is to use the current sensor data measurements and compare them against the map. The Filter-based estimation technique can help in identifying the wrong prediction in the second step, if there is any, then it returns to the previous step. The repetition steps give update for each new measurement. These two steps can be considered as check and balance between an estimation calculation and determining the vehicle's pose position acquired from the sensor data measurements from the map in real time [8]. Similar to the Filter-based category, the Optimization-based category also has two parts. The first part is to detect the constraints of the problem, in this case the vehicle's current pose, from the sensor data measurements. This can be achieved by finding the connection between the new sensor data measurements and the map to achieve a unified whole of the estimation of the vehicle's poses with respect to the map [8].

2.1 Kalman Filter

The Kalman Filter (KF) was proposed by Kalman in 1960. KF can be applied to orbit calculation, target tracking and navigation as it can work in real time, fast, efficient and has strong anti-interference. It uses series of data observed over time to estimate unknown variables with more accuracy. KF is a linear optimal status estimation method [7]. The current position is predicted based on the previous position features of the environment. The difference in actual and estimated observation is then calculated. The Kalman gain is then calculated and the predicted position is corrected. Finally, the new environmental feature is added to the map.

2.2 Extended Kalman Filter

Meanwhile, the Extended Kalman Filter (EKF) derived from KF adds a linearization step which is commonly used in non-linear filtering. However, EKF covariance matrices are quadratic in the number of landmarks, and in order to update them, the time quadratic in the number of landmarks is needed [9]. EKF assumes that the state transition and the measurements are Markov processes represented by non-linear functions [10]. EKF is unable to support large-scale maps that are continuously growing because it requires time in a quadratic way. To overcome this issue, the notion of submaps was created [1],[3] and every time the map gets too big, a new blank map will be replaced. All the submaps will be kept tracked by a higher-level map.

2.3 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) was introduced to compensate the weaknesses of the EKF with highly non-linear systems, which avoid the computation of the Jacobians. The idea of this approach is to sample particles called sigma points that are pondered around the expected value thanks to a likelihood function. These sigma points are then passed on to the non-linear function and the estimate is recomputed. The major drawback of this method is its computational time. Most of the study and research that use the UKF algorithm took place at the beginning of the 2000s [3],[12].

2.4 Karto SLAM as a Graph SLAM

Three parts of the graph Optimization-based are motion estimation, graph optimization and loop closure detection [8]. Many localization problems can be modelled using graph representation. Matrix between the vehicle poses and the landmarks can be built easily and used in optimization framework. An example of the Graph SLAM method or approach is the Karto SLAM. Karto SLAM is a graph-based SLAM approach proposed by SRI International Karto Robotics, which has been plugged in to ROS by using a highly optimized and non-iterative Cholesky matrix decomposition [12].

A graph-based SLAM algorithm represents the map with each node is a pose of the robot along its trajectory and a set of sensor measurements. These nodes are connected by arcs for the motion between successive poses. For each new node, the computation of the map is done by finding the spatial configuration of the nodes that are consistent with constraints from the arcs [13]. In the Karto SLAM in ROS, the Sparse Pose Adjustment (SPA) is responsible for both scan-matching and loop-closure procedures [12]. More amount of memory is required when the number of landmarks become higher. However, graph-based SLAM algorithms are usually more efficient in maintaining a map of large-scale environments. Meanwhile, Karto SLAM is extremely efficient since it only maintains a pose graph.

Obstacle avoidance in navigations of a mobile robot played an important role [12],[14] and the robot should be able to collect the data from attached sensors and process information to make the right decisions based on the surrounding environment of the robot that are dynamic. Obstacle avoidance algorithm may have advantages and disadvantages depending on the environment that the robot located, outdoor or indoor, the shape of the obstacle and the performance of the mobile robot [14], [16]. Obstacle avoidance in known environment is less complicated than in unknown environment. The information of the environment including the obstacles' position is provided to the robot rather than the robot needs to map the unknown environment itself when the robot is positioned into the known environment [17],[18].

3. METHODOLOGY

This case study is done through the Ubuntu 16.0.4 platform in Intel Core i7 machine with 8GB RAM and ROS Kinetic with Turtlebot package for robot simulation. ROS software provides two visualization platforms. The first platform is called Gazebo, which provides the robot simulation. This platform is used in creating the environment where the simulation robot can be tested. The second visualization platform is the RViz where it can visualize the robot's laser scan and other features such as the visualization of the possible path of the robot. The main difference that could be seen between these two platforms, aside from the different functions, is that the Gazebo visualize in 3D while the RViz visualize in 2D. Other than that, RViz functions as the mapping of environment.

After the ROS Kinetic is installed, the next step was to setup the working environment in it. In other words, is to setup a catkin workspace to be the main workspace that allows multiple projects to be built in it as well as the built-in packages from ROS through Turtlebot simulation. The following table 1 shows the commands and steps to install Turtlebot in ROS environment with localization model which is the Karto SLAM algorithm.

Table 1: Steps of setting the environment in ROS

	Steps
1	Open environment model
2	Run Karlo SLAM algorithm
3	Visualize the map trhough RViz
4	Control simulation robot through keyboard
5	Drive the simulation robot
6	Mapping the environment
7	Save the map

The project makes use of the robot simulation to move the model and the localization algorithm used is the Karto Localization. The process flow of is shown in Figure 2, which starts with the given initialized map, where the robot moves around and data from its sensors is captured and a new map is produced. At the same time, the robot with sensor scans any barriers and move again by avoiding the obstacles.

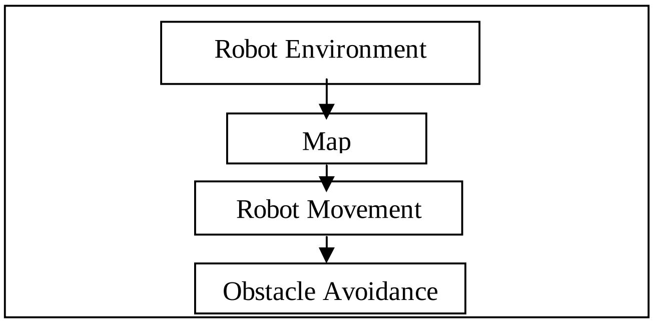


Figure 2: The flow of processes in the experiment

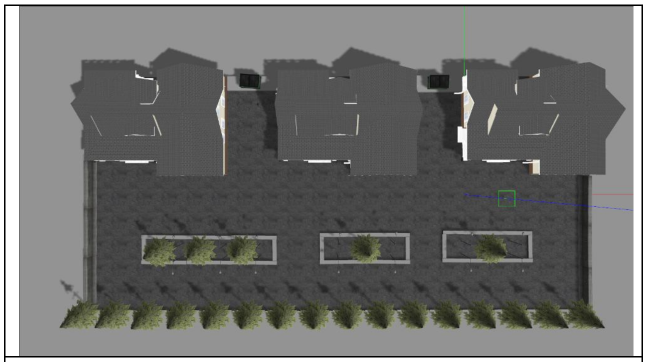
After the setting of the environment is completed, the robot model was tested and evaluated in two different simulation environments to ensure it was working and that the output given was accurate and not contrary to the actual results. The result testing was visualized by ROS simulation. The testing and evaluation were repeated until the localization model gave satisfactory results without any significant error(s). The performance of the simulation model was recorded after the testing and validation processes were completed and the accuracy of the robot model were analysed to show how well the model worked. Next, the environment was equipped by distance measures in the obstacle avoidance algorithm. Therefore, the improvement that have been made is notification of distance through popup warning.

3. RESULTS AND DISCUSSION

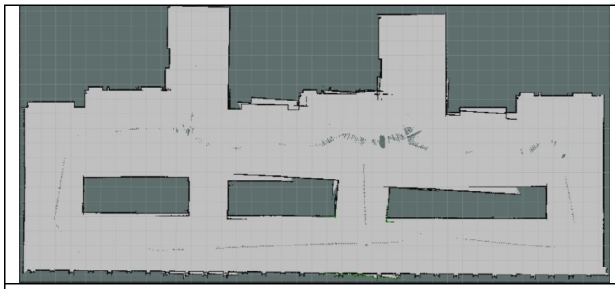
In simulating the localization of the Turtlebot robot, two environments were created to test the accuracy of the simulation robot. From the created environments, the map of each environment was built based on Karto SLAM and Gmapping. Gmapping is one of the default mapping methods that was provided by ROS. Two environments were provided, which are house and maze. House contains houses and a few objects such as mailbox, dumpster, pine tree and lamp post. The second environment that was created consisted of walls that were arranged to produce a maze. The maze was constructed to test the localization of the simulation robot to navigate itself towards its goal which is to escape from the maze.

3.1 Comparison of Map Produced between Karto and GMapping SLAM

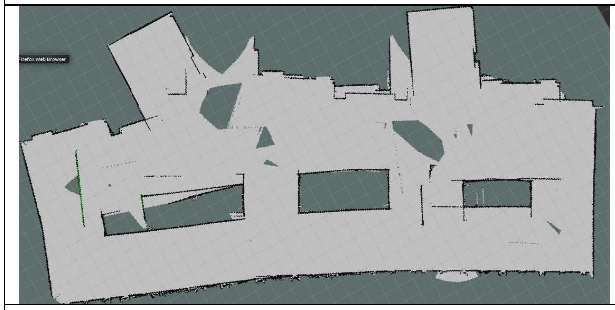
In this project, the maps generated using GMapping and Karto algorithm were visualized and compared. Figures 3 and 4 show the map produced using Karto SLAM and GMapping for house and maze. Maps produced were displayed in RViz, a 3D visualization tool that comes together with a full ROS desktop installation.



a) House environment

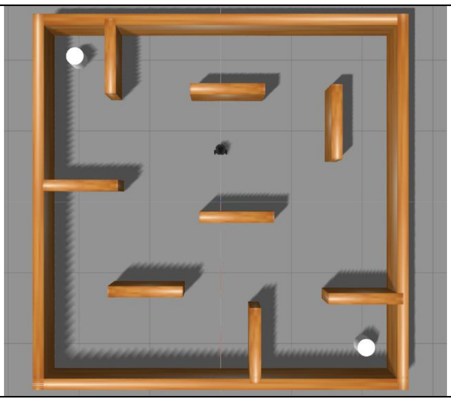


b) House environment map using Karto SLAM

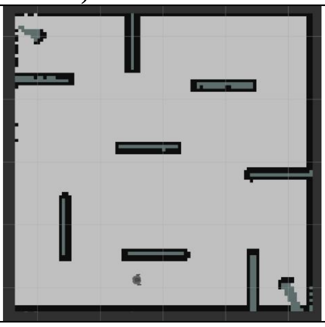


c) House environment map using GMapping

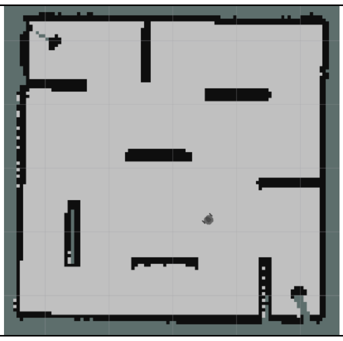
Figure 3: House Environment a) for original view b) Map using Karto SLAM and c) Map using GMapping



a) Maze environment



b) Maze environment map using Karto SLAM



c) Maze environment map using GMapping

Figure 4: Maze Environment a) for original view b) Map using Karto SLAM and c) Map using GMapping

The comparison of the map could be seen clearly from Figures 3 and 4. From this experiment, Karto SLAM provided a better map compared to GMapping. This is because maps provided by using Karto SLAM are similar to the real environment especially for the house environment. On the other hand, the map that was produced by using Gmapping algorithm shows some anomalies in which the map is slightly slanted as compared to the real environment for the house environment. Therefore, it can be summarized that Karto SLAM can map large-scale and complex environment well compared to GMapping.

Figures 5 and 6 show the CPU and memory usage when using Karto SLAM and GMapping. Based on these figures, GMapping tends to use more CPU and memory resources compared with Karto SLAM. This is because five (5) out of eight (8) show that CPU usages are higher when running GMapping algorithms. In addition , GMapping also tends to use higher ram consumption at 2.2 GB out of 7.7 GB RAM, with Karto SLAM using a ram consumption of 2.1 GB out of 7.7 GB RAM when running the simulation.

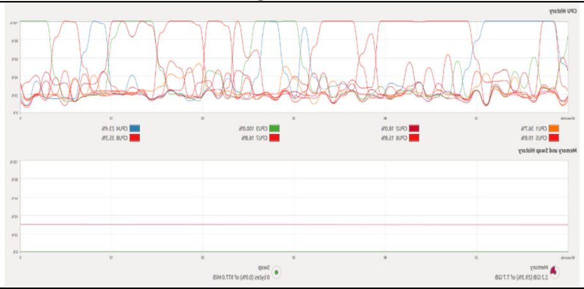


Figure 5: CPU and Memory Usage for Karto SLAM

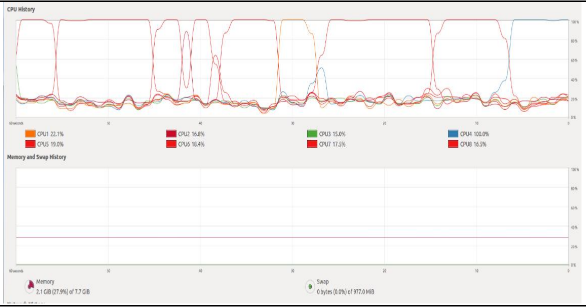


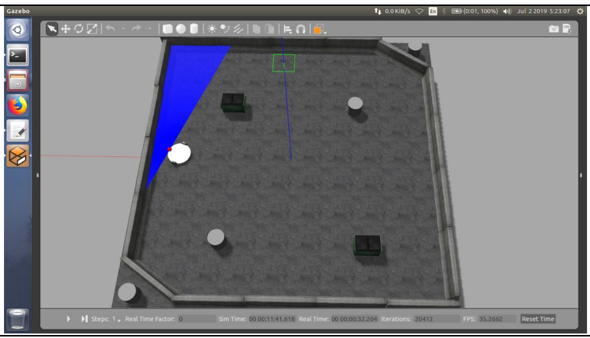
Figure 6: CPU and Memory Usage for GMapping

3.2 Application of Obstacle Avoidance Notification

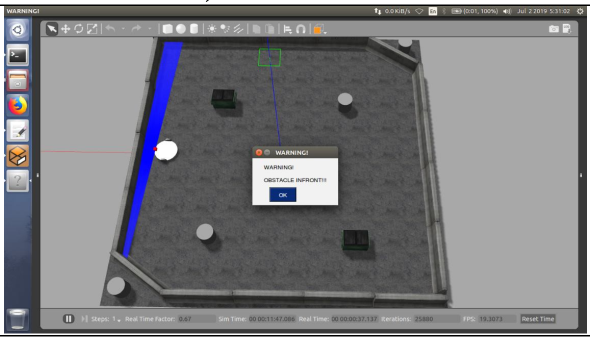
The robot is equipped with Hokuyo 2D Sensor module. The sensor has 180° field-of-view blue coloured rays. The sensor module is located on the robot that are red coloured box-shaped. The sensor is capable of reaching 10 meters in range of the 180° field-of-view. The improvement made to the simulation was done by calculating the distance of the robot to the obstacle when the obstacle is aligned in a straight line from the robot. This means that the robot will avoid any obstacle that are in its path. Next, the popup warning that will alert the robot on the nearby obstacle with certain threshold. Each experiment was run for 3 times each for the original and the improved simulation. The tested environments were as below:

- Several objects are placed in certain region in the environment. Two cylinders and two dumpsters are placed near the corner of the environment.
- In the second environment, the robot is expected to walk around and avoid the wall that are placed in irregular pattern.
- The third environment is based on maze environment with walls. The robot should be able to start from the beginning point, green coloured arrow, following the white line and finish at the end point, red coloured arrow.

In robot obstacle avoidance algorithm, the maximum gap is assigned 40 meters which is the minimum distance that robot maintains with obstacle not to avoid a collision. Range angles refer to list of range angles the robot can rotate which can vary from - 0.5(anticlockwise) to threshold value 1.5. A threshold value is a maximum limit of range angle set for robot to move through the map.



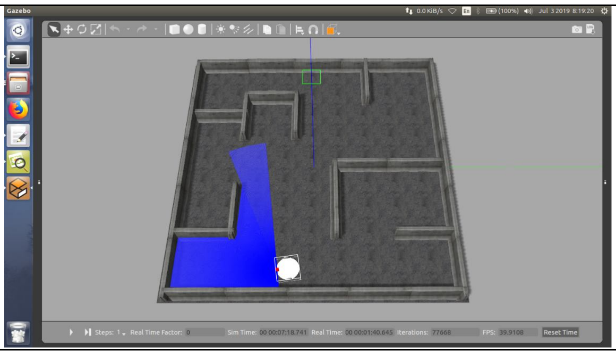
a) Without notification



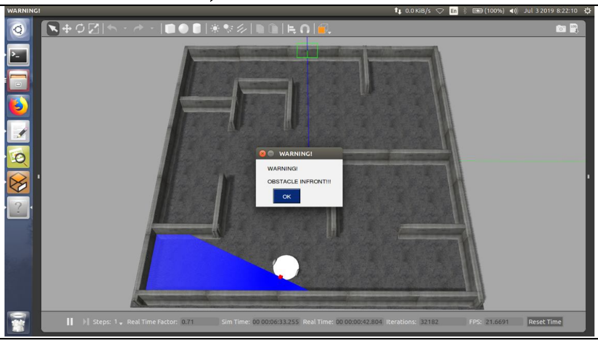
WARNING!
WARNING!
OBSTACLE INFRONT!!!
OK

b) With notification

Figure 7: Environment 1



a) Without notification



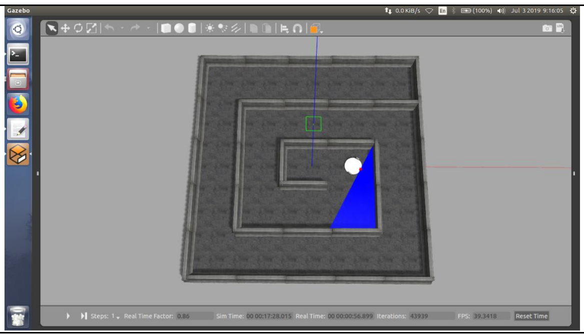
WARNING!

WARNING!
OBSTACLE INFRONT!!!

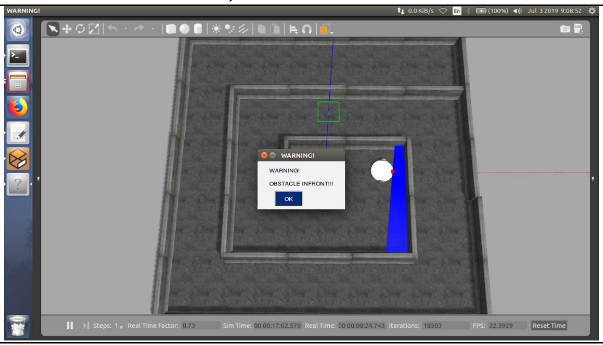
OK

b) With notification

Figure 8: Environment 2



a) With notification



WARNING!

WARNING!
OBSTACLE INFRONT!!!

OK

b) With notification

Figure 9: Environment 3

Figure 7, 8 and 9 are the environments which equipped without and with the notification. As can be seen from each of the environment testings, the module could produce popup warning to the user without fails. The results can be concluded that the improved module is successful and can be applied to other researches and projects.

4. CONCLUSION

In simulating the localization of the Turtlebot robot, two environments were created to test the accuracy of the simulation robot. From the created environments, the map of each environment was built based on Karto SLAM and Gmapping. Gmapping resulted in a simulated world was promising since it could correct itself when the robot arrived back at the known environment. Overall, Karto SLAM gave better mapping results and the use of computer resources is comparatively better. The popup warning is also provided when the distance of the robot is near to the obstacle. This project shows a promising future that can be applied to improve the performance of the autonomous mobile robot and the autonomous vehicle. For further research, the use of real Turtlebot to test the accuracy in a real world can be undertaken and can be compared with other algorithms [19, 20] to test which one among them is better.

REFERENCES

1. A. D. M. Africa and C. F. C. Uy, **Development of a Cost-Efficient Waste bin Management System with Mobile Monitoring and Tracking**, *International Journal of Advanced Trends in Computer Science and Engineering*, Volume 8 (2), pp. 319-327, 2019. https://doi.org/10.30534/ijatcse/2019/35822019
2. G. G. Kalach and G. P. Kalach, **Navigation System Based on the Fuzzy Logic Expert System**, *International Journal of Advanced Trends in Computer Science and Engineering*, Volume 8 (6), pp. 2693 – 2698, 2019. https://doi.org/10.30534/ijatcse/2019/02862019
3. L. Wang, Y. Zhang and J. Wang. **Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR**. *IFAC-PapersOnLine*, Vol. 50(1), pp. 276–281, 2017.
4. S. Abdul-Rahman, M. S. Abd Razak, A. H. Mohd Mushin, R. Hamzah, N. Abu Bakar, Z. Abd Aziz, **Simulation of simultaneous localization and mapping using 3D point cloud data**, *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 14(2), pp. 1–9, 2018.
5. A. Pajaziti and P. Avdullahu, **SLAM – Map Building and Navigation via ROS**, *International Journal of Intelligent Systems and Applications in Engineering (IJISAE)*, Vol. 2 (4), pp 71-75, 2014. https://doi.org/10.18201/ijisae.08103
6. A. Mohd Azri, S. Abdul-Rahman, R. Hamzah, Z. Abd Aziz, N. Abu Bakar, **Visual analytics of 3D LiDAR point clouds in robotics operating systems**, *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 9, No. 2, pp 492–499, 2020.
7. Q. Li, R. Li, K. Ji and W. Dai, **Kalman Filter and Its Application**, *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Tianjin, 2015, pp. 74-77.

8. T. T., Takleh Omar, N. Bakar, S. Abdul-Rahman, R. Hamzah and Z. Abdul Aziz**. A Brief Survey on SLAM Methods in Autonomous Vehicle**, *International Journal of Engineering &Technology*, Vol. 7 No 4.27, pp. 38–43, 2018. https://doi.org/10.14419/ijet.v7i4.27.22477
9. S. Kamijo, Y. Gu and L. Hsu. Autonomous Vehicle Technologies : **Localization and Mapping**, *IEICE ESS Fundamentals Review*, Vol. 9, issue 2, pp. 131–141, 2015.
10. L. Zhang, R. Zapata and P. Lépinay, **Self-adaptive Monte Carlo localization for mobile robots using range sensors**, *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, 2009, pp. 1541-1546.
11. D. Fox, S. Thrun, W. Burgard and F. Dellaert, **Particle Filters for Mobile Robot Localization**. In: Doucet A., de Freitas N., Gordon N. (eds) Sequential Monte Carlo Methods in Practice. *Statistics for Engineering and Information Science*. Springer, New York, NY, pp. 401-428, 2001. https://doi.org/10.1007/978-1-4757-3437-9_19
12. S. J. Julier and J.K. Uhlmann Julier, **Unscented Filtering and Nonlinear Estimation**, *Proceedings of the IEEE (Volume: 92 , Issue: 3 , March 2004)*, pp. 401 – 422, 2004.
13. J. M. Santos, D. Portugal and R. P. Rocha, **An evaluation of 2D SLAM techniques available in Robot Operating System,** *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Linkoping, pp. 1-6, 2013.
14. C. Giannelli, D. Mugnaini and A. Sestini. Path planning with obstacle avoidance by G^1 PH quintic splines. **Computer-Aided Design**, Vol. 75–76, pp. 47–60, 2016.
15. N. Kumar, and Z. Vámossy. **Obstacle recognition and avoidance during robot navigation in unknown environment**. *International Journal of Engineering & Technology*, Vol. 7(3), pp. 1400, 2018. https://doi.org/10.14419/ijet.v7i3.13926
16. P. D. I. Torino. **Obstacle Avoidance Algorithms for Autonomous Navigation system in Unstructured Indoor areas**. M. S thesis, Politecnico Di Torino, 2018.
17. Y. Chen and J. Sun. **Distributed optimal control for multi-agent systems with obstacle avoidance**. *Neurocomputing*, Vol. 173, pp. 2014–2021, 2016.
18. E. A. Wan and R. Van Der Merwe, The unscented Kalman filter for nonlinear estimation, Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373), Lake Louise, Alberta, Canada, 2000, pp. 153-158.
19. Z. Zakaria, N. Nordin, A. Md Ab Malik, S. J. Elias, Ahmad Z. Shahuddin, **Fuzzy expert systems (FES) for halal food additive**. *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 13 (3), March 2019.
20. A. N. Fadzal, M. Puteh and N. Abd Rahman, Ant colony algorithm for text classification in multicore-multithread environment, Indonesian Journal of Electrical Engineering and Computer Science, Vol 18, No 3: June 2020, pp1359-1366. https://doi.org/10.11591/ijeecs.v18.i3.pp1359-1366