Indonesian Journal of Electrical Engineering and Computer Science
Vol. 20, No. 1, October 2020, pp. 500~509
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v20.i1.pp500-509

Developing a real time navigation for mobile robots at unknown environments

Sarah Haider Abdulredah, Dheyaa Jasim Kadhim
Department of Electrical Engineering, University of Baghdad, Iraq

Article Info

Article history:

Received Jan 7, 2020
Revised Mar 5, 2020
Accepted Mar 19, 2020

Keywords:

EKF-SLAM
Gazebo and rviz
Mobile robot
ROS
Turtlebot2e

ABSTRACT

Mobile robot needs to navigate at unknown environments and constructing its maps at the same time. Therefore, we proposed to use an algorithm named simultaneous localization and mapping (SLAM). Then, we suggested the extended kalman filter algorithm (EKF) to solve the SLAM problem which is implemented at different unknown environments containing a different number of landmarks where the detectable landmarks will play an important role in controlling the overall navigation process and on EKF-SLAM technique's performance. MATLAB simulation results show that the performance of EKF-SLAM path is enhanced as the number of landmarks increased, so the performance becomes better as compared with an odometry path depending on the value of mean square error. After that, we simulated mobile robot platform named TurtleBot2e in Gazebo simulator to achieve the SLAM algorithm for different environments based on G-mapping algorithm which was built on robot operating system (ROS). The main contribution that comes with this work is the simulation of SLAM technique is done by using two different software platforms separately (MATLAB and ROS). Finally, the execution time to build a map is computed for each environment in Gazebo simulator, and we concluded that it is increased when the landmarks are increased.

Corresponding Author:

Sarah Haider Abdulredah,
Department of Electrical Engineering,
University of Baghdad, Baghdad, Iraq.
Email: sarah.haider94@yahoo.com

1. INTRODUCTION

The latest research trends in robotics have witnessed dramatic advancements in using robots to facilitate human difficult jobs such as the survey of mars or analyses of the sea bottom such applications need that the robot can navigate in unknown environments [1-7]. The navigation issue is considered one of the most important issues in current research trends in robotics [8-11]. A key condition for successful navigation is the capability of localization and building maps of unknown environments for a mobile robot, which is also well-known as simultaneous localization and mapping (SLAM) [12, 13]. Firstly, SLAM is presented in 1986 which is considered as one of the AI techniques that benefit from the mobile robot's navigation in at unknown environment [14], to construct a new map for this environment so as other robots can navigate it smoothly in the future [15]. So, the SLAM problem can define as a case study of a chicken-or-egg problem [16-18].

The main problem with SLAM is that measurements that read from the sensors always containing noise and uncertainties in its positioning produced by the motion of the robot [19, 20]. So, there are different solutions to solve the SLAM problem. The best estimation approaches that used such extended kalman filter (EKF) [21], unscented kalman filter (UKF) [22] and rao blackwellised particle filter (RDPF) [23]. In our work, we will focus on the Extended Kalman Filter algorithm [24]. EKF-SLAM is dependent on the EKF algorithm which is an extended approach that is used originally for estimation of the linear systems while the EKF is used for estimation of the non-linear systems [12, 25-28].

Briefly, we can summarize our work into two phases at the first phase, our proposal modified EKF-SLAM is simulated and implemented using MATLAB simulator for different unknown environments that contain different numbers of landmarks to study the effect of the increasing number of landmarks on EKF-SLAM path and Odometry path and to enhance the performance of the mobile robot trajectory. At the second phase of this research work, we suggest a mobile robot named TurtleBot2e to achieve the SLAM algorithm in real simulation world called by Gazebo world using G-mapping algorithm which is the package in ROS provides the tools to create a 2D base map using laser and data of Odometry [29-31]. In this work, the Turtlebot2e robot is equipped with an IR emitter sensor provided by the Kinect camera to detect the environment which contains different numbers of landmarks and obstacles [32-34].

There are several research works are developed in the last years ago that deal with SLAM, EKF-SLAM and their applications in robotics environments as follows: In [24], described the implementation of Extended Kalman Filter utilizing Python over a mobile robot and investigation of its qualification in the actual existence of the physical environment In [25], 2D EKF-SLAM for mobile wheeled robots is applied to evaluate the state of a UGV operating in a real environment involving dynamic objects in a real environment. In [31], introduced a simulation environment for mobile robots based on ROS and Gazebo. In [32], presented the SLAM that is implemented in ROS by calculating the travel time taken by the robot model to reach the endpoint. In [33], presented the SLAM algorithm based on ROS for simulated mobile robot on Gazebo simulator with a scanning laser that is introduced with PR2 and AR-601M robots. In [34], presented the SLAM framework based on low-cost LiDAR and vision fusion is implemented by using Turtlebot2 where the navigation package provided by the ROS.

So, our main contribution of this work is the simulating and achieving of the SLAM algorithm separately in MATLAB and ROS software platforms for different unknown environments having a different number of landmarks which they will play an important role in determining the mobile robot's location and constructing the corresponding maps of these environments for future usage in addition to improve the performance of SLAM technique. The ROS simulation result shows a few numbers of landmarks will make the mobile robot loses its path; therefore, increasing the number of landmarks was given a more accurate path. Finally, the mobile robot needs a long time to observe the unknown environment.

2. PROPOSED PROCEDURE

In our work, we suggested using EKF as a filter to solve the SLAM problem which is one of the greatest famous methods utilized in the localization procedure of mobile robots where the problem occurs due to the noise and uncertainties in its positioning produced by the motion of the robot. So, EKF-SLAM is simulated and implemented using MATLAB. Then, we suggested enhancing the performance of EKF-SLAM by increasing the number of landmarks. Then, we increased the number of landmarks to see the effect of increasing on the performance of EKF-SLAM trajectory and Odometry trajectory and we watched what happened to the path of the robot in the event of an increase or decrease in the number of landmarks. So, EKF uses first-order Taylor series extension to grasp the linearization of nonlinear. To describe the EKF-SLAM process, it consists of two stages as follows.

2.1. EKF-SLAM prediction stage

In this stage, we consider only the previous estimated positions of the Odometry motion model to predict the current position of the mobile robot. It can be expressed by mean and covariance which the predicted mean $\bar{\mu}_t$ is equal to nonlinear function g depended on the previous mean μ_{t-1} and control u_t as in (1) the covariance matrix $\bar{\Sigma}_t$in (1) is taken into account the Jacobian matrix of nonlinear function g. It is depending on the previous covariance matrix, and R_t which is defined as noise covariance that is added to process through the motion control as follows:

$$\bar{\mu}_t = g(\mu_{t-1}, u_t), \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \qquad (1)$$

The dimension of EKF-SLAM is (2n+3). So, to map the nonlinear function g according to the dimension (2n+3) space. The state of the mobile robot will change but the state of landmarks are not changed. Now, we will compute the Jacobian matrix by taking the partial derivative for nonlinear function g according to the robot and landmarks positions. The error ellipse will be large, uncertainty will grow and the effect of left and right controller becomes smaller in the prediction stage. The solution is to go to use the correction stage of EKF.

2.2. EKF-SLAM correction stage

In this stage, we consider the correction gain is necessary to update the mean and covariance of the current state. Also, it is known as the Kalman gain. Now, we compute the Kalman gain K_t by compute the the Jacobian function $H_t$of observation function h. The computation of Kalman gain tells us how much the certainty the robot about its predicted belief and how much the certainty of the robot about sensing a property of observation, and then we have:

$$K_t = \overline{\sum}_t H_t^T \left(H_t \overline{\sum}_t H_t^T + Q_t \right)^{-1} \tag{2}$$

Q_t is the uncertainty of the sensor observation (noise covariance). Finally, the mean and covariance of the correction stage of EKF-SLAM are produced as follows:

$$\mu_t = \bar{\mu}_t K_t \left(z_t - h\left(\bar{\mu}_t\right)\right), \Sigma_t = \left(I - K_t H_t \right) \overline{\Sigma}_t \tag{3}$$

3. RESEARCH METHODOLOGY

Using MATLAB simulator, we proposed three types of mobile robot trajectory such as actual trajectory, EKF-SLAM trajectory and Odometry trajectory that move in a circular path counter clockwise at the proposed environment; there will be an intersection among the taken measurements. Now, because of the circular features of the proposed environment, there will be an intersection among the taken measurements. So, it is difficult to see the locations of these three categories at each time instance and this causes an inability to notice the efficiency and performance of the proposed EKF-SLAM technique and odometry. To solve this issue, we go to determine the distance error between the actual robot trajectory to EKF-SLAM robot trajectory denoted by (D_1) and distance error between the actual robot trajectory to odometry robot trajectory denoted by (D_2) as follows:

$$D1 = \surd\left(xEst - xActual \right)^2 + (yEst - yActual)^2 \tag{4}$$

$$D2 = \surd\left(x\ Odo - x\ Actual \right)^2 + (y\ Odo - y\ Actual)^2 \tag{5}$$

$\left(x\ Actual, y\ Actual \right), \left(x\ Est, y\ Est \right)$ and $(x\ Odo, y\ Odo)$ are the coordinates of the actual, estimated and odometry robot positions respectively. To study the effectiveness of increasing the number of landmarks on the performance of both EKF-SLAM and Odometry trajectories, we have to determine the mean square error for both previous distance errors D_1 and D_2 by MSE_1 and MSE_2 respectively where MSE_1 is the mean square error of the distance error between the actual robot trajectory to EKF-SLAM robot trajectory and MSE_2 is the mean square error of the distance error between the actual robot trajectory toOdometry robot trajectory, the expressions of these two metrics are given by:

$$MSE1 = \sum_{i=1}^{k} D_1 \ \ and \ \ MSE2 = \sum_{i=1}^{k} D_2 \tag{6}$$

where k is the number of robot locations. Now, numbers of experiments are tested in MATLAB with a different number of landmarks. According to following expressing, we compute the performance of EKF-SLAM which it is denoted by *Perf.1* and Odometry performance which it is denoted by *Perf.2*:

$$Perf.1 = \left(1 - MSE1\right) \times 100, \ and \ Perf.2 = \left(1 - MSE2\right) \times 100 \tag{7}$$

3.1. Model design and implementation

In this part of our work, we proposed to use Gazebo simulator to simulate the mobile robots navigation at unknown environment using SLAM technology. For practical considerations, we adopted here to use a mobile robot platform named TurtleBot2e to do our tests in Gazebo simulator with Rviz library built on ROS in Linux. In our tests, we simulated the Gazebo world that contains different numbers of landmarks to execute the moving of mobile robots at an unknown environment. So, the environment is built in the Gazebo world that contains the same number of landmarks that are tested in the previous environments in MATLAB. In this work, the mobile robot starts to navigate different unknown indoor environments which contain different number of landmarks built-in Gazebo simulator to test the effect of increasing the number of landmarks on the path of the mobile robot and how much of time that is required to estimate its position and construct a map of these unknown environments, where the SLAM algorithm is achieved using the g-mapping tool. In this work, The mobile robot start to navigate in different unknown indoor environment which contains different number of landmarks built-in Gazebo simulator to test the effect of increase the number of landmark on the path of the mobile robot and on the time which taken to localize itself by observe these landmark and construct a map of these unknown environment. So, SLAM algorithm is achieved.

3.1.1. Installing ROS

We start to install the robotic operating system (ROS) on an Ubuntu in Linux OS which is provided on the website of ROS.org with obvious steps evidence to complete the installation. In this work, the version of Ubuntu is 16.04 LTS is used. Regarding the ROS with the latest distribution, the Kinetic is the codename, which is compatible with Ubuntu 16.04. ROS is installed in Ubuntu using terminal software, in which various precautions were taken to start the installation.

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc)
main" > /etc/apt/sources.list.d/ros-latest.list'
```

The following step is to type the next command in terminal to setting up the key.

```
wget https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -O - |
sudo apt-key add -
```

After the source list and configuration keys are defined, the procedure of installing as well as downloading ROS is completed. So, firstly we create sure that the newest Debian set is set up by update the ROS packages by writing the following command.

```
sudo apt-get update
```

The next step is the ROS installation package download process. To download this package, we have three diverse choices. In the first method is a full download of the installation ROS package.

```
sudo apt-get install ros-kinetic-desktop-full
```

This involved 2D/3D simulation programs, 2D/3D navigation and insight, the Rviz 3D visualization instrument, a graphical interface progress instrument named Rqt and generic robot archives. The installation of the basic desktop is the second choice which individual involved Rviz, Rqt and the robot-generic libraries.

```
sudo apt-get install ros-kinetic-desktop
```

The installation of ROS-bare bone is the third choice that involved only the packages of ROS, and the libraries of builds and communication.

```
sudo apt-get install ros-kinetic-ros-base
```

The complete office installation is selected for this job. After selecting and installing the installation package, rosdep had to be initialized to allow easy installation of system dependencies as well as operate some basic mechanisms in the Robot Operating System.

```
sudo rosdep init
rosdep update
```

A new shell is started at each time it is suggested to add ROS environment variables to the bash sitting each time. This is complete by issuing the command.

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

It is also suggested that set up the commonly used rosinstall command-line tool. With rosinstall, we can easily download many sources for the packages of ROS with a single command. Syntax is provided in the following command to download rosinstall.

```
sudo apt-get install python-rosinstall
```

After following the installation instructions, ROS is ready for the Ubuntu operating system.

3.1.2. Installing TurtleBot2e packages
In ROS, the execution of automatic navigation with the personal robot TurtleBot2e is tested in a simulated world. The TurtleBot2e package can be installed with this command from the terminal.

```
$ sudo apt-get install ros-kinetic-turtlebot ros-kinetic-turtlebot-apps ros-kinetic-turtlebot
interactions ros-kinetic-turtlebot-simulator ros-kinetic-kobuki-ftdi ros-kinetic-roconremocon ros-
kinetic-rocon-qt-library ros-kinetic-ar-track-alvar-msgs
```

3.1.3. Building environments in gazebo
Then, we build an unknown environment with a number of features by opening a gazebo simulator window and building it inside Gazebo and saving it on Linux with a specific name. Then we call the TurtleBot2e inside this environment through the following command:

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch world_file:=worlds/home/sarah.world name
```

We tested the simulation of the TurtleBot2e which starts to navigate in an unknown environment. The robot is of the opinion that he has no idea of his environment and therefore cannot yet navigate freely. Using any laser and odometer data from the mobile robot, a map can be calculated and created by using the g-mapping algorithm built using the ROS tool

3.1.4. Controlling TurtleBot2e at unknown environment
We allow the TurtleBot2e to move everywhere in the environment physically, the remote keyboard control is required to control the TurtleBot2e. The terminal command line has been executed to move the mobile robot.

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

The basic movements of TurtleBot2e are controlled by the keyboard of our PC. Finally, to increase the speed of the mobile robot is click on q-keyboard. In this work, the speed of the TurtleBot2e is reached to 0.7m/s.

3.1.5. Starting G-mapping tool
We make an internal environment map by starting the g-mapping instrument. The command line in the terminal has been specified. The g-mapping package supplies Simultaneous Localization and Mapping (SLAM), as ROS node named SLAM g-mapping.

```
$ roslaunch turtlebot_gazebo gmapping_demo.launch
```

3.1.6. Visualizing using Rviz platform
The 2-D map progress visualization is visible from the Rviz platform. So, the Rviz platform is used to view the navigation of the mobile environment when build a map. The terminal command line is run to view the navigation of TurtleBot2e.

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

3.1.7. Saving YAML-file

The map made from SLAM g-mapping process is kept via the command line in the terminal where the file of the map is saved as a YAML-file.

```
$ rosrun map_server map_saver -f <map name>
```

The extension "my_map" in the previous command can be any name we assign to the map that was built on Rviz. While my_map.yaml is a short file that contains information about the map image but also has the fixed name of the image file. So, if we alternate the map file name, be sure to edit the yaml file to make the corresponding name alternate, then also alternate the yaml file name itself.

3.1.8. Requesting saved map

In the previous last step, the map is saved for use at another time, then to allow the mobile robot to navigate in this map of the known environment. The gazebo world, which we built its map, is re-opened by returning the command line in the step 4 in the second window of a command-line is entered in the terminal below to open the final map.

```
roslaunch turtlebot_gazebo amcl_demo.launch map_file:=/home/sarah/map name.yaml
```

3.1.9. Viewing navigation process

Finally, a command-line is entered in the terminal below to view the navigation of the mobile robot in Known environment.

```
$ roslaunch turtlebot_rviz_launchers view_navigation.launch
```

4. RESULTS AND DISCUSSION

Firstly, in phase.1 different environment containing one landmark and eight landmarks organized in a circular path with a different location (x, y) and then, test the impact of the increasing landmark from one to eight on the performance of the EKF-SLAM path and Odometry path. The varying of the distribution form of landmarks inside the environment is no impact on the EKF-SLAM performance these environments can be expressed as shown in Figures 1 and 2. So, these environments are simulated in MATLAB software using M-file to test the effectiveness of the EKF-SLAM trajectory and odometry trajectory according to the actual robot trajectory we see the EKF-SLAM trajectory (desired path) is closed to the actual path while the odometry trajectory diverges from the actual trajectory for one landmark case and become more closed to actual trajectory when the number of landmarks is increased to eight landmarks

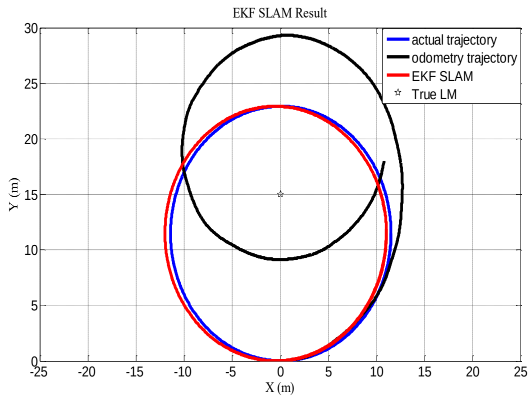

Figure 1. The environment contains one landmark

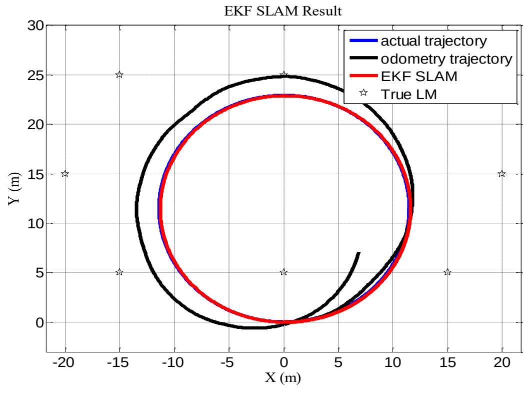

Figure 2. The environment contains eight landmarks

The distance errors D1 and D2 are obtained for the trajectories of EKF-SLAM as shown in Figure 3 where MSE1 and MSE2 were computed for both D1 and D2. The value of MSE1 is lower than MSE2 therefore, the performance in the environment which contains one landmark is not perfect but the performance of the environment that have eight landmarks is very perfect and becomes higher as compared to the previous cases that have few landmarks so the distance errors of the environment that contain eight landmarks to test the performance of EKF-SLAM is shown in Figure 4.

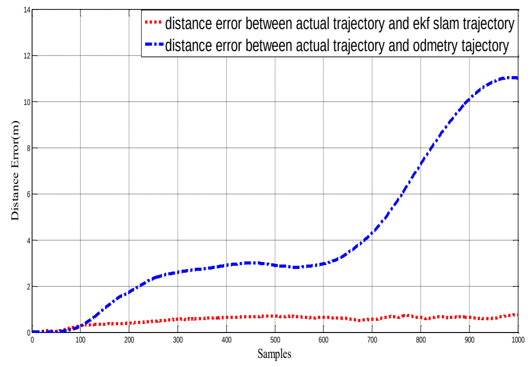

Figure 3. D1 and D2 of environment contain one landmark

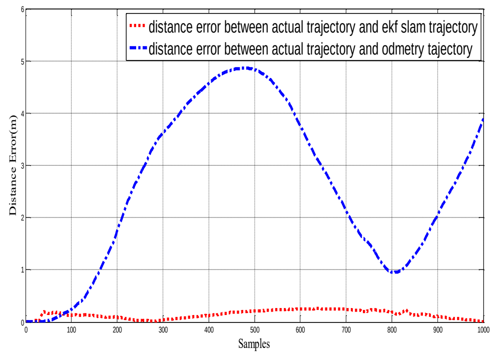

Figure 4. D1 and D2 of environment contains eight landmark

Other experiments of EKF-SR SLAM have been tested in MATLAB with different numbers of landmarks to show the effect of increasing the number of landmarks on the effectiveness of the EKF-SLAM trajectory and compared it with Odometry robot trajectory which included in Table 1.

Table 1. The MSE and Per. of both EKF-SLAM path and odometry path

Number of landmarks	M_1	M^2	MSE_1 (m)	MSE_2 (m)	Per.1	Per.2
1	0.5559	7.1234	0.4023	4.2773	59.77	-327.7
3	0.3254	6.4562	0.3797	5.3838	62.03	-438.4
4	0.3334	5.5525	0.2956	2.2808	70.44	-128.2
5	0.2678	3.2918	0.2468	3.6542	75.32	-364.4
6	0.4509	7.0416	0.2034	4.0978	79.66	-309.7
8	0.1595	6.1.234	0.1723	5.1876	82.77	-517.7

Now, we start to describe the results of the previous environments. Table 1 shows the MSE1 values of EKF-SLAM is decreased from 0.4023 to 0.1723 when the number of landmarks is increased from one to eight landmarks and also, the performance is increased from 59.77 to 82.77. At whole six environments of EKF-SLAM when we compare the performance of EKF-SLAM with the Odometry performance we see the performance of EKF-SLAM (Perf.1) is higher than the performance of Odometry (Perf.2) especially when the number of landmarks is increased. The performance of the environment three that have four landmarks is more perfect than the performance in the environment two that includes three landmarks and the performance is become higher to be 82.77 as compared to the previous case when the number of landmarks is increased from one to three then to eight. In the second phase, we evaluated a real-time SLAM algorithm in the ROS platform for our proposed and we used TurtleBot2e as a robot to navigate the environment models that are created using Gazebo. We have placed cylindrical and blue landmarks in several different locations within the Gazebo simulator and the simulation is performed under the ROS control. We suggested increasing the number of landmarks to study the effect of increasing the number of landmarks on the movement and path of the robot. Figures 5 and 6 showing the interior environment that contains different landmarks starting from one landmark to eight landmarks where the TurtleBot2e is attached Kinect camera with IR emitter sensor to observe and scan landmarks in the unknown environment. The TurtleBot2e starting to observe all the landmarks in each environment and we suggest computing the time for the robot to complete its constructing map.

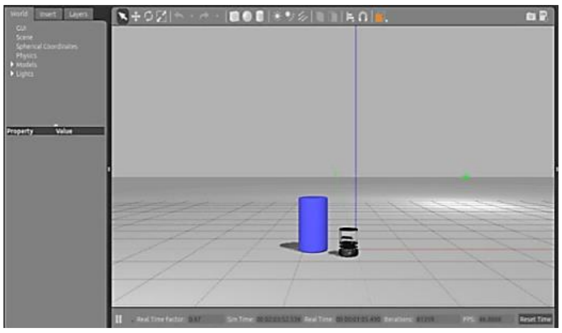
Figure 5. The gazebo world which contains one landmark

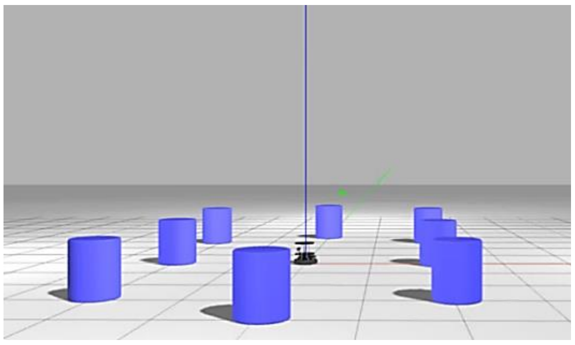
Figure 6. The gazebo world with eight landmarks

The TurtleBot2e control procedure at known environment is used to move everywhere to the created map. Firstly, the navigation begins by selecting the (2D Pose Estimate) button and placing the green arrow at the position and orientation of the TurtleBot2e on the map. In a second or two we will see the cloud of points from AMCL move to the location followed by the TurtleBot2e model jumping to that location. We may do this any number of times before creating a navigation goal. Secondly, select the (2D Nav Goal) button and click on a target location and orientation to start the navigation. This process may be repeated as long as batteries hold. The TurtleBot's global path plan and local path plan will be visible on the map as the TurtleBot2e navigates. Finally, the planned route along with the TurtleBot2e location to the destination's endpoint is displayed then the robot will start moving towards the destination that is selected where the robot can navigate anywhere in the final map using these tools. Figures 7 and 8 showing the navigation in the final map (known environment) after the robot observed all landmarks at each environment using the Rviz tool and from the results, we see with the number of landmarks are increased to eight landmarks the mobile robot can estimate its location easily to become more accuracy than the environment which contains one landmark because the mobile robot can allocate itself with a number of landmarks, also we see when the number of landmarks is increased, we obtain an accurate path for mobile.

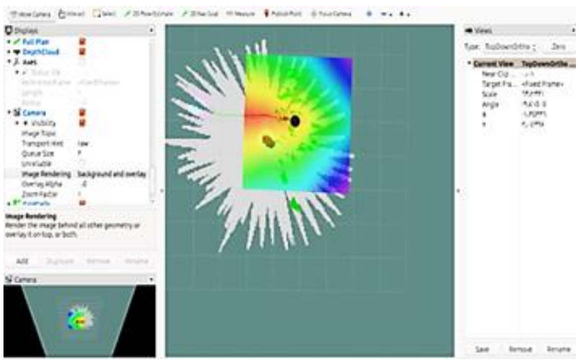
Figure 7. The navigation on an environment map that contains a one landmark using rviz tools

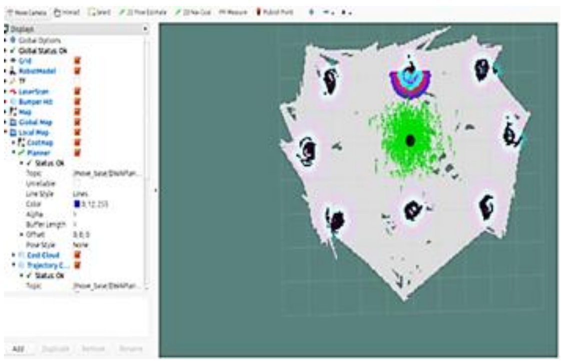
Figure 8. The navigation on environment map contains eight landmarks

Then, we will take the time factor as an influencing factor when building an unknown environment in ROS. From the results of the implementation of the previous six environments, we see that when the number of landmarks is increased, the robot takes more time to build its environment in Table 2. We see the real-time that needed to build the unknown environment is increased from 6.42 minutes to 24.21 minutes when the number of landmarks is increased from one to eight as shown in Figure 9. Therefore, the mobile robot takes a lot longer time when the robot entered into a difficult environment that contains a lot of fixed landmarks for this reason, the robot after it completes the task of building the map of the environment that it entered. It stores the final map. So, we have to suggest that the cloud to be used to store the completed maps for later use by the robot itself or by another robot that wants to complete a task with the fastest time. Instead of losing time to build a map of the place that enters a mechanism therefore, robot uses stored map through connecting to the cloud.

Table 2. The real execution time vs. number of landmarks

No. of landmarks	Real time/minute
1	6.42
3	13.43
4	16.33
5	19.14
6	20.19
8	24.21

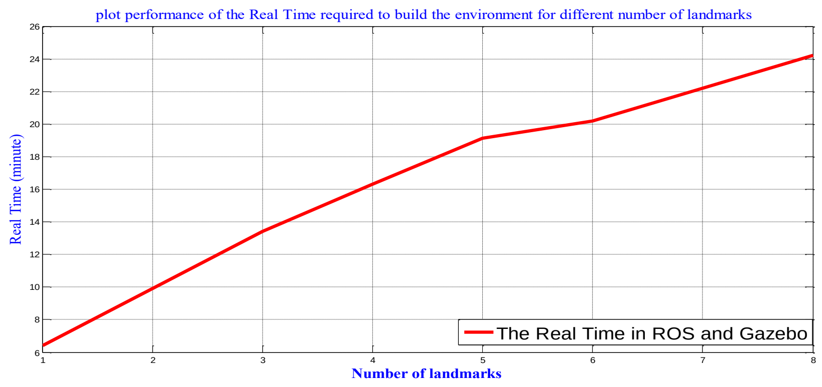

Figure 9. The real time required to build different unknown environment

5. CONCLUSION

The first approach proposed in this work is to solve the SLAM problem by using extended kalman filter (EKF) through two stages (prediction and correction stages). Different environments that contain different landmarks are simulated using MATLAB. The MATLAB simulation results showed that the performance of the proposed EKF-SLAM is enhanced as the number of landmarks is increased. The performance of EKF-SLAM for the environment that contains one landmark is 59.77, while the performance of EKF-SLAM for the environment that contains eight landmarks is 82.77. Also, from the aggregated results, EKF-SLAM approaches gave an improvement to the path of mobile robot as compared to the Odometry path according to the actual path of mobile robot, where the performance of EKF-SLAM trajectory outperforms than the performance of Odometry trajectory. This work implemented and simulated SLAM algorithm in different indoor environments contain different number of landmarks using Gazebo simulator on ROS based on g-mapping algorithm. A few numbers of landmarks will make the mobile robot loses its path, and it will take a long time to observe all environment in Gazebo world when the mobile robot was entered in the unknown environment to obtain mapping, therefore, increasing the number of landmarks is more accuracy when compare it with environment which contains few numbers of landmarks. The time required to achieve SLAM on ROS is increasing when the number of landmarks is increasing also.

REFERENCES

[1] R. Siegwart, et al., "Introduction to autonomous mobile robots," *2nd ed. London: The MIT press*, 2011.
[2] B. S. Ranbir, et al., “Internet of robotic things: Driving intelligent robotics of future-concept, architecture, applications and technologies,” *2018 4th International Conference on Computing Sciences (ICCS)*, Jalandhar, pp. 151-160, 2018.
[3] S. Sundarajoo and A. S. A. Ghani, "Improvement of auto-tracking mobile robot based on HSI color model," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 12, no. 3, pp. 1349-1357, 2018.
[4] A. L. Robert and W. G. Culllen, "Work in progress: Introductory mobile robotics and computer vision laboratories using ROS and MATLAB," in *ASEE Annual Conference and Exposition, Conference Proceedings*, United States, 2018.
[5] H. Widyantara, et al., "Wind direction sensor based on thermal anemometer for olfactory mobile robot," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 13, no. 2, pp. 475-484, 2019.
[6] F. Affane, et al., "Type-2 fuzzy logic controller optimized by wavelet networks for mobile robot navigation," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 18, no. 1, pp. 326-334, 2020.
[7] C. Lamini, et al., "Genetic algorithm based approach for autonomous mobile robot path planning," *Procedia Computer Science*, vol. 127, pp. 180-189, 2018.
[8] S. Ahmed, "Design of a cognitive neural predictive controller for mobile robot," PhD thesis, Dep. Electronic and Computer Engineering, Brunel Univ., United Kingdo, 2012.

[9] D. S. Kumar, et al., "Development of Path Planning Algorithm of Centipede Inspired Wheeled Robot in Presence of Static and Moving Obstacles Using ModifiedCritical-SnakeBug Algorithm," *International Journal of Artificial Intelligence (IAES)*, vol 8, no. 2, pp. 95-106, 2019.

[10] J. S. Pershina, et al., "Methods of Mobile Robot Visual Navigation and Environment Mapping," *Optoelectronics, Instrumentation and Data Processing*, vol. 55, no. 2, pp. 181-188, 2019.

[11] W. R. Abdulmajeed, et al., "Implementing Autonomous Navigation Robot for building 2D Map of Indoor Environment," *International Journal of Computer Applications*, vol. 92, no. 2, pp. 7-13, 2014.

[12] X. Baiyu, et al., "Simulation research for active simultaneous localization and mapping based on extended kalman filter," *2008 IEEE International Conference on Automation and Logistics*, Qingdao, pp. 2443-2448, 2008.

[13] M. R. Naminski, "An analysis of Simultaneous Localization and Mapping (SLAM) Algorithms," *Macalester College Mathematics, Statistics, and Computer Science, Honors Projects*, vol. 29, 2013.

[14] H. M. Hasan and T.H. Mohammed, "Implementation of mobile robot's navigation using SLAM based on cloud computing," *Engineering and Technology Journal*, vol. 35, no. 6, pp. 634-639, 2017.

[15] K. Jajulwar, et al., "Design of Mobile Robot Navigation system using SLAM and Adaptive Tracking Controller with Particle Swarm Optimization for Indoor Environment Monitoring," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 17, no. 6, pp. 59-63, 2015.

[16] A. Hamzah, et al., "A hypothesis of state covariance decorrelation effects to partial observability SLAM," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 14, no. 2, pp. 588-596, 2019.

[17] D. A. Baramiya, et al., "Simultaneous Localization and Mapping System on the Basis of a CoreSLAM Approach," *Optoelectronics, Instrumentation and Data Processing*, vol. 53, no. 6, pp. 599-603, 2017.

[18] A. Esa, et al., "Simulation of simultaneous localization and mapping using hexacopter and rgbd camera," *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, Jakarta, pp. 48-53, 2017.

[19] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 56-68, 1986.

[20] M. Hanheide, et al., "Exploiting probabilistic knowledge under uncertain sensing for efficient robot behaviour," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 2442-2449, 2011.

[21] N. Ayadi, et al., "Simulation and experimental evaluation of the EKF simultaneous localization and mapping algorithm on the wifibot mobile robot," *Journal of Artificial Intellegent Soft Computing Research (JAISCR)*, vol. 8, no. 2, pp. 91-101, 2018.

[22] M. S. Bahraini, et al., "A new adaptive UKF algorithm to improve the accuracy of SLAM," *International Journal of Robotics*, vol. 5, no. 1, pp. 35-46, 2018.

[23] L. Maohai, et al., "Novel mobile robot simultaneous loclization and mapping using rao-blackwellised particle filter," *International Journal of Advanced Robotic Systems*, vol. 3, no. 3, pp. 231-238, 2006.

[24] A. B. S. H. M. Saman and A. H. Lotfy, "An implementation of SLAM with extended Kalman filter," *2016 6th International Conference on Intelligent and Advanced Systems (ICIAS)*, Kuala Lumpur, pp. 1-4, 2016.

[25] R. P. Saputra, "Implementation 2D EKF SLAM for Wheeled Mobile Robot," Master thesis, School of Mechanical and Manufacturing Engineering, Unsw Australia, 2015.

[26] R. Mur-Artal, et al., "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transaction on Robotics*, vol. 31, no. 5, pp. 1147-1163, 2015.

[27] E. Jakob, et al., "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conference on Computer Vision, 2014, ECCV 2014*, Zurich, Switzerland, vol. 8690, pp. 834-849, 2014.

[28] K. Stefan, et al., "A Flexible and Scalable SLAM System with Full 3D Motion Estimation," *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, Kyoto, pp. 155-160, 2011.

[29] E. L. Voisan, et al., "ROS-Based Robot Navigation and Human Interaction in Indoor Environment," *2015 IEEE 10th Jubilee International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, pp. 31-36, 2015.

[30] P. Y. Sang and L. H. Gui, "Mapping and Localization of Cooperative Robots by ROS and SLAM in Unknown Working Area," *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, Kanazawa, pp. 858-861, 2017.

[31] T. Kenta, et al., "Simulation environment for mobile robots testing using ROS and Gazebo," *2016 20th International Conference on System Theory, Control and Computing (ICSTCC)*, Sinaia, pp. 96-101, 2016.

[32] R. K. Megalingam, et al., "ROS based autonomous indoor navigation simulation using SLAM algorithm," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 7, pp. 199-205, 2018.

[33] A. Ilya, et al., "ROS-based SLAM for a Gazebo-simulated mobile robot in image-based 3D model of indoor environment," in *International Conference on Advanced Concepts for Intelligent Vision Systems*, vol. 9386, pp. 273-283, 2015.

[34] G. Jiang, et al., "A Simultaneous Localization and Mapping (SLAM) framework for 2.5 D map building based on low-cost LiDAR and vision fusion," *Applied Sciences*, vol. 9, no. 10, pp. 1-17, 2019.