Performance of SLAM Algorithms According to Noise and Reduced Landmark Parameters

Ismail Bartal1, Ahmet Özkurt2

[1]The Graduate School of Natural and Applied Sciences, Dokuz Eylul University, Izmir, TURKEY
ismail.bartal@ogr.deu.edu.tr
[2]Department of Electrical and Electronics Engineering, Dokuz Eylul University, Izmir, TURKEY
ahmet.ozkurt@deu.edu.tr

Abstract

The aim of this study is to test SLAM algorithms in the Webots simulation program and compare the performance results according to several noise levels and landmark conditions. Simulations have carried out using a robot with a differential driving system. The 2D lidar sensor on the robot has used to detect obstacles in the environment. Predetermined landmarks placed in the simulation environment to enable the robot to estimate its position using different SLAM algorithms. The location information obtained with different number of landmarks has compared and the results show that the position estimation error increases when the number of landmarks decreases. Subsequent studies aimed to increase the accuracy of location estimation by creating landmarks using the points obtained from the Lidar sensor.

1. Introduction

A robot must perform simultaneous localisation and mapping(SLAM) in order to move autonomously in an unknown environment. The robot needs location information to perform mapping. Likewise, the robot needs map information in order to perform positioning operations. As discussed by Cadena [1], this makes the SLAM problem a complex one.

As discussed by Stachniss [2], there are three different solution methods for solving the SLAM problem. These methods are Kalman filter, particle filter and graph-based approach. In this study, we have used the Extended Kalman Filter (EKF) using the Kalman filter method and the Fast SLAM solution methods using the particle filter method.

As discussed by different works [3], [4], [5] Kalman filter (KF) works on linear systems. But a real system is generally nonlinear. For this reason, the extended Kalman filter (EKF) is preferred method. The EKF algorithm filters the erroneous result by linearizing non-linear systems. EKF SLAM consists of two stages, which have discussed in the section 2.2.

The particle filter is an alternative approach for solving the SLAM problem. As shown by Murphy et all. [6], this method uses particles that contain the pose, map and weight information of the robot. Fast SLAM approach has developed by Montemerlo et all. [7]. Each of the particles used in the Fast SLAM algorithm has a Kalman filter. Each particle tries to estimate the landmark locations using the Kalman filter.

For a robot to move autonomously, it needs to plan a path. During path planning, the robot tries to arrive at the target location while avoiding the obstacles around it. In this study, pathes have created to points with known locations so that the robot can move autonomously.As shown by Fox et all. [8] Dynamic windowing approach (DWA) are uses in the path planning process. In this method, vectors are created according to the robot's speed and angular velocity information. These vectors are evaluated based on the location of the surrounding obstacles and the target position. The most appropriate vector is selected as a result of the evaluation process. The velocity and angular velocity information of the selected vector is given to the robot as control input. The robot moves with the given control information. This process is repeated while the robot is moving.

Many SLAM algorithms have been developed today. Each algorithm has different advantages and disadvantages. Therefore, there is a need to compare between algorithms in order to select the SLAM algorithm that gives the desired result. As shown by Kümmerle [9] A method has developed to compare SLAM algorithms metrically. In the method used, the robot's position and angle information are evaluated simultaneously. However, in this study, position and angle information have evaluated separately.

2. Material and Methods

In this section, the introduction of the Webots simulation program, the structure of EKF SLAM and the structure of Fast SALM have presented respectively.

2.1. Webots Simulation Program

A controlled environment is needed to test the algorithms used in the SLAM problem. This makes it easier to run and develop the SLAM problem, which is difficult to develop. Webots simulation program has used in this study. As shown in [10], the Webots simulation program can be used to The environment for the SLAM application can be created, the desired robot can be designed, the movement of the robot and the data received from the sensors can be monitored instantly, It facilitates error detection, It shortens the development process because the development of the algorithm is done in a controlled environment.

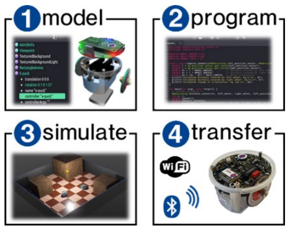

Fig. 1. Structure of the Webots simulation program [11].

In this study, EKF SLAM and Fast SLAM algorithms have compared using Webots simulation program. A robot with a differential driving system have designed in the simulation program. Also an environment with corners and straight walls have designed. Red boxes were placed in the environment designed for SLAM algorithms to estimate the location.

Figure 2 shows the environment designed in the Webots simulation program. The red boxes represent landmarks with known locations, the white and black obstacles represent walls, the red cylinder represents the robot with the differential drive ystem, the green cylinder represents the desired target location, and the light blue circle represents the area visible to the lidar sensor.

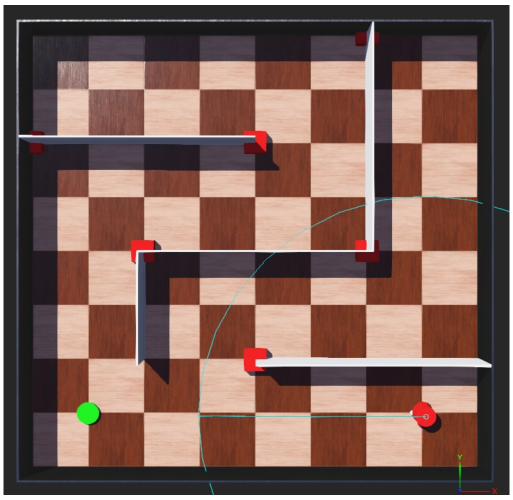

Fig. 2. Designed environment in Webots simulation program.

In this study, the robot does not take into account the obstacles around it when observing landmarks. Landmarks are set to be observable even if they are behind a wall. In this way, as soon as the landmarks enter the field of view of the lidar sensor, they send their location information to the robot.

2.2. EKF SLAM

The EKF SLAM algorithm uses an extended Kalman filter to filter out noise from the sensors and the environment. Unlike the Kalman filter, the EKF can operate in nonlinear systems.In this way, it can better solve the SLAM problem in a real situation.

The EKF SLAM algorithm basically consists of two steps. These steps are respectively state estimation and state updating. In the estimation stage, the next state is tried to be estimated according to the previous sensor data and the position of the robot. In the updating stage, the estimated robot position and map information are compared with the actual values received from the sensors. As a result of the comparison process, the Kalman gain is calculated and tries to bring the situation closer to the real result as described in [3], [4].

The EKF SLAM algorithm does not record all past states. It only estimates current position information and landmarks. This SLAM structure is called Online SLAM [5].The EKF SLAM algorithm does not record all past states. It only estimates current position information and landmarks. This SLAM structure is called Online SLAM [5].

As shown in Figure 3, EKF SLAM uses the Online SLAM solution method. It estimates the current position(x) and map(m) according to the given control signal(u) and observation(z) data.

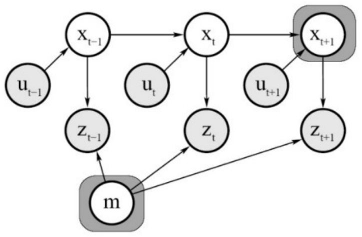

Fig. 3. Online SLAM

2.3. Fast SLAM

The Fast SLAM algorithm tries to solve the SLAM problem using a particle filter. The particle filter basically consists of 4 steps. These steps are particle generation, state estimation, state update and resampling.

As discussed [6], [7], in the Fast SLAM algorithm, particles are randomly distributed in a certain number around the robot. Each particle has different position, rotation and map information. As the robot moves with the applied control information, each particle also moves. As the robot moves, it tries to estimate the location and map information. The actual position obtained from the sensors and the predicted position according to the map information are compared. If particles are close to the true value, their weight increases, while particles which are not close to the true value disappear over time. As the number of particles decreases, new particles are generated by resampling around the particle with the highest weight.

Fast SLAM saves all past positions. It tries to estimate the current position and map based on the past positions. This SLAM structure is called full SLAM [12].

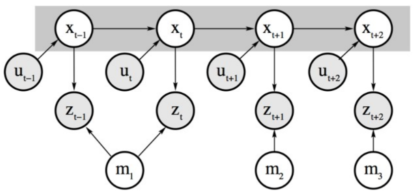

Fig. 4. Full SLAM

Figure 4 shows a graphical representation of the full SLAM structure. In the Full SLAM approach, it calculates the position and map with the given control(u) and observation(z) information. Unlike the online SLAM structure, it takes into account all past positions when calculating the current position.

As seen in Figure 4, in the Fast Slam algorithm, there is no connection between landmarks(m) since the landmarks are independent of the robot's posture [12].

3. Simulation Result

In this section, EKF SLAM and Fast SLAM algorithms will be tested in Webots simulation program respectively. The testing process consists of 4 stages. The first step is to compare the graphs of the robot positions obtained without adding noise to the control input and sensors. In the second stage, the robot positions obtained by adding 5% noise to the control input and sensors will be compared with metric error measurement methods. In the third stage, using EKF and Fast SLAM algorithm, the process will be repeated by reducing the number of landmarks in the case of 5% noise. In the fourth stage, the landmarks will be completely removed and the robot positions estimated by the SLAM algorithms will be compared with metric error measurement methods.

In the Webots simulation program, SLAM algorithms were created using the python programming language. In the simulation process, the duration of each cycle is set to 64 milliseconds. The simulation environment shown in Figure 2 is a square with a side of 2 meters. The robot used in this environment has a differential drive system. The lidar sensor on the robot has used to detect obstacles in the environment.

During the simulation, DWA was used for the robot to move autonomously. In this method, a green cylinder is placed in the simulation so that the robot can move to the desired locations while avoiding the obstacles around it. This cylinder represents the target position for DWA. When the robot reaches the target position, the target position is automatically moved to the next determined position. In this way, the robot moved autonomously in the created environment.

3.1. Simulation with Noiseless Sensor Data

In order to better evaluate the results of the SLAM algorithms in the presence of noise, all sensor noise in the simulation is set to zero in this section.

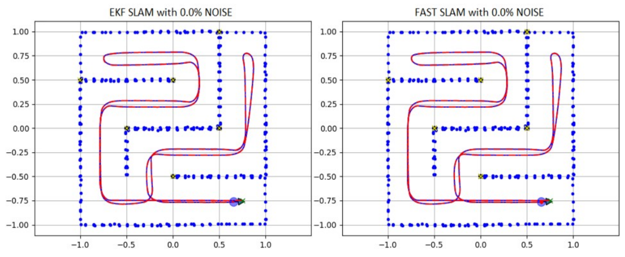

Fig. 5. EKF SLAM(Left) and Fast SLAM(Right) with noiseless sensor

Figures 5 show the location and map obtained with the SLAM algorithms in the noiseless case. The light blue circle represents the robot. The blue line shows the actual robot position and the robot position calculated with the encoder sensor, while the red line shows the robot position estimated with the SLAM algorithm. Blue dots indicate the map obtained according to the real robot position, black asterisks indicate the actual position of the landmarks in the simulation environment, and yellow crosses indicate the landmark positions estimated by the EKF and Fast SLAM algorithms as a result of observing the landmarks with noise-free lidar data.

As can be seen in Figures 5, the robot position estimated by the SLAM algorithms in a noiseless case is exactly the same as the actual robot position.

3.2. Simulation with Noisy Sensor Data

In this section, 5% noise is applied to the control input and sensors. For the noisy case, EKF and Fast SLAM algorithms are used to estimate the position of the robot. Figures 6 show the location and map obtained with the SLAM algorithms in 5% noise. The robot positions obtained with EKF and Fast SLAM have recorded during the simulation period. The simulation have stopped when the robot traveled around the whole area prepared in the simulation and returned to its starting position.

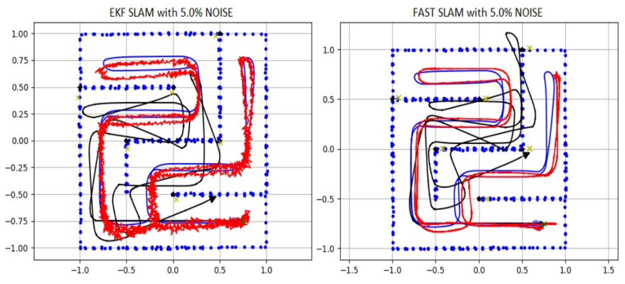

Fig. 6. EKF(Left) and Fast(Right) SLAM with 5% noise

When the robot returns to its initial position, the error value obtained by using all actual robot positions recorded and the predicted robot position is calculated by metric error measurement methods.

$$MSE = \frac{1}{N} \Sigma_{i=1}^{N} (x_i - X_i) \tag{1}$$

$$RMSE = \sqrt{\frac{1}{N} \Sigma_{i=1}^{N} (x_i - X_i)^2} \tag{2}$$

$$MAE = \frac{1}{N} \Sigma_{i=1}^{N} |x_i - X_i| \tag{3}$$

MSE (Mean Squared Error), RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error) values in equations (1), (2) and (3) were calculated according to x, y and rotation values. where x_i represents the i-th measured true value, X_i represents the i-th predicted value and N represents the number of measurements.

Table 1. MSE values obtained from 5% noise simulation

5% Noise	Mean squared error (MSE)		
	MSE_X	MSE_Y	MSE_ROT.
EKF SLAM	0.00114	0.000579	0.00114
Fast SLAM	0.00770	0.00050	0.00770

Table 2 . RMSE values obtained from 5% noise simulation

5% Noise	Root mean squared error(RMSE)		
	RMSE_X	RMSE_Y	RMSE_ROT.
EKF SLAM	0.03375	0.02406	0.03375
Fast SLAM	0.08776	0.02236	0.08776

Table 3. MAE values obtained from 5% noise simulation

5% Noise	Mean absolute error(MAE)		
	MAE_X	MAE_Y	MAE_ROT.
EKF SLAM	0.02918	0.01796	0.02918
Fast SLAM	0.08403	0.01717	0.08403

As seen in Tables 1, 2 and 3, each axis was evaluated separately when calculating the position error in this study. As can be seen in the tables, the EKF SLAM algorithm gave less inaccurate results than the Fast SLAM algorithm. Only in the y-axis, the Fast SLAM algorithm predicts the robot position with less error than the EKF SLAM algorithm.

3.3. Simulation with Reduced Landmark Count

At this section, 5% noise is applied to the control input and sensors. At the same time, 2 landmarks were removed from the simulation environment. The selection of these two landmarks have based on the range of the lidar sensor. Before the number of landmarks have reduced, the lidar sensor could see up to 5 landmarks at the same time in the simulation environment. Similarly, when the number of landmarks is reduced, it can see at most 3 landmarks at the same time. For the SLAM algorithm to estimate the robot position correctly, it must see at least 2 landmarks continuously. This number was obtained as a result of simulations. When the 2 selected landmarks are removed from the simulation environment, the lidar sensor can see at least 1 landmark.

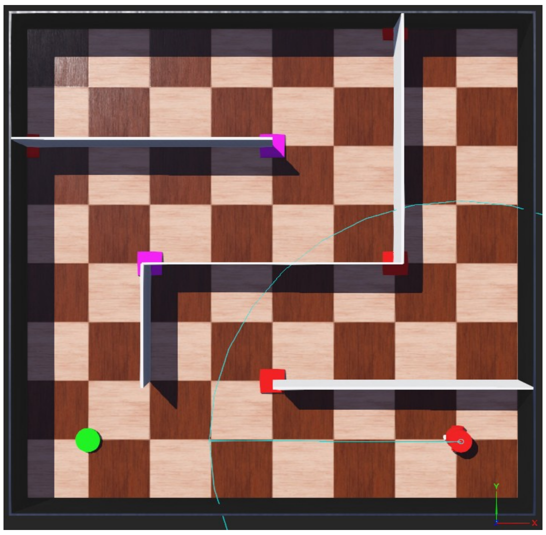
Fig. 7. Landmarks extracted in Webots simulation program (Purple squares)

Figure 7 shows the landmarks extracted in the Webots simulation program. These points have selected from the places where the robot made the most turns. If the robot turns without seeing enough landmarks, there is more error in the estimated robot position. Tables 4, 5 and 6 show the error values obtained as a result of reducing the number of landmarks and metric error measurement methods. In section 3.2, the position error in the y-axis is calculated less when the number of landmarks is not reduced. When the number of landmarks is decreased, the error in the y-axis increases for both EKF SLAM and Fast SLAM algorithms. The error increase in the y-axis is higher for the EKF SLAM algorithm and lower for the Fast SLAM algorithm.

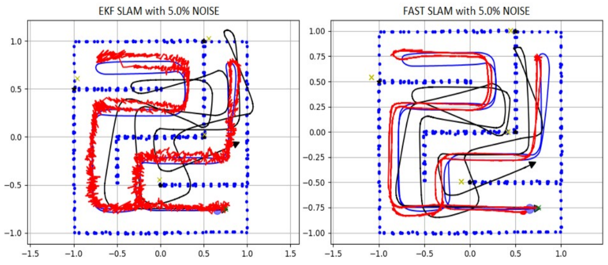

Fig 8. EKF(Left) and Fast(Right) SLAM with reduced number of landmarks

Table 4. MSE values obtained by reducing the number of landmarks

Reduced Number of Landmarks	Mean squared error (MSE)		
	MSE_X	MSE_Y	MSE_ROT
EKF SLAM	0.00106	0.00418	0.00106
Fast SLAM	0.00781	0.00103	0.00781

Table 5. RMSE values obtained by reducing the number of landmarks

Reduced Number of Landmarks	Root mean squared error(RMSE)		
	RMSE_X	RMSE_Y	RMSE_ROT.
EKF SLAM	0.03262	0.06461	0.03262
Fast SLAM	0.08839	0.03213	0.08839

Table 6. MAE values obtained by reducing the number of landmarks

Reduced Number of Landmarks	Mean absolute error(MAE)		
	MAE_X	MAE_Y	MAE_ROT.
EKF SLAM	0.02627	0.05402	0.02627
Fast SLAM	0.08441	0.02168	0.08441

3.4. Simulation with all landmarks removed

To better understand the importance of the number of landmarks in SLAM algorithms, all landmarks in the simulation were removed. At the same time, 5% noise was added to the sensors. In the simulation without landmarks, the robot position estimated by the SLAM algorithms has compared with the actual position information. Metric error measurement methods have used in the comparison process.

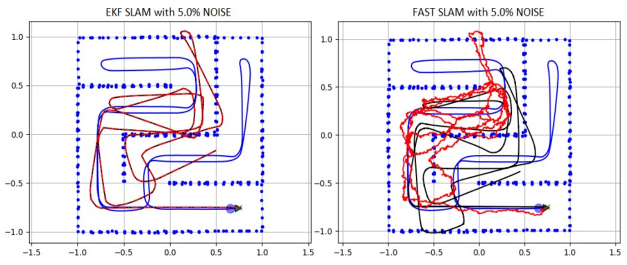

Fig 9. EKF(Left) and Fast(Right) SLAM with 5% noise and no landmark

Table 7. MSE values obtained without landmark

Without Landmark	Mean squared error (MSE)		
	MSE_X	MSE_Y	MSE_ROT.
EKF SLAM	0.04447	0.06721	0.04447
Fast SLAM	0.13314	0.11066	0.13314

Table 8. RMSE values obtained without landmark

Without Landmark	Root mean squared error(RMSE)		
	RMSE_X	RMSE_Y	RMSE_ROT.
EKF SLAM	0.21089	0.25925	0.21089
Fast SLAM	0.36489	0.33266	0.36489

Table 9. MAE values obtained without landmark

Without Landmark	Mean absolute error(MAE)		
	MAE_X	MAE_Y	MAE_ROT.
EKF SLAM	0.13671	0.21977	0.13671
Fast SLAM	0.25679	0.24598	0.25679

Tables 7, 8 and 9 show the position errors of the SLAM algorithms without using landmarks with metric error measurement methods. When the errors calculated in Section 3.3. are compared, it is seen that the error values increase. Fig. 9 show the graph of the simulation using EKF and Fast SLAM algorithms. The EKF SLAM algorithm estimated the same position as the encoder data due to the absence of landmarks. However, the Fast SLAM algorithm cannot fully evaluate the encoder data because it uses a particle filter. This is because in the particle filter, the particles need landmark information during the resampling process.

4. Conclusion

In this study, four different simulations were performed using EKF SLAM and Fast SLAM algorithms in the Webots Simulation program. As a result of these simulations, it has observed that the errors in location estimation increased with decreasing the number of landmarks. However, when the noise is reduced, the location estimate is closer to the true value. Simulations have shown that noise and number of landmarks significantly affect the location accuracy in SLAM algorithm

In a noiseless situation, even if the SLAM algorithms accurately estimate the true robot position, this does not reflect reality. In a real situation, there will be noise from the environment and sensors. These noises vary according to the quality of the environment and the sensor. In this study, 5% noise was added to the sensors to analyze a noisy situation. The reason for choosing 5% noise is that the errors are more visible.

MSE, RMSE and MAE methods have used for comparison. With these methods, the position error has calculated separately for each axis. As a result of the simulation performed in Sections 3.2, 3.3 and 3.4, it is seen that the error in the y-axis is higher than the other axes. In order to better evaluate this situation, simulations on different maps have aimed.

EKF and Fast SLAM algorithms need landmarks to estimate the position of the robot. When landmarks are not used, SLAM algorithms are not able to reduce the accumulated error. The simulation results show that Fast SLAM requires more landmarks for location estimation.

In Sections 3.2 and 3.3, a simulation process close to a real situation has performed. In these simulations, it was seen that the number of landmarks affects the estimated position of the robot. Both EKF SLAM and Fast SLAM algorithms gave results close to the true location with different numbers of landmarks. The EKF SLAM algorithm achieved less location error with a given number of landmarks.

In this study, landmark extraction was not performed with lidar sensor since the landmarks were already placed in the simulation environment. Noise added during the observation of landmarks have not applied to the lidar sensor on the robot. This is not the case with a lidar sensor operating in a real environment. In addition, the robot moved in a partially known environment as it observed landmarks whose locations were predetermined. However, SLAM algorithms can also perform location and mapping in an unknown environment.

In future studies, it is aimed to develop a landmark extraction algorithm using data from lidar sensor in a completely unknown environment. In order to develop this algorithm, noise will be added to the lidar sensor at a determined value within the webots simulation program. This noise value will be based on a real lidar sensor. In this way, the lidar sensor will produce more realistic results in the simulation. However, the noise introduced to the lidar sensor will also negatively affect the landmark extraction algorithm. Therefore, it is aimed to develop a landmark extraction algorithm that works in no landmark conditions.

4. References

[1] C. Cadena, L. Carlone, H. Carrillo, ark, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age." *IEEE Transactions on robotics, Vol(*32), 1309-1332, 2016.

[2] C. Stachniss, J. J. Leonard, S. Thrun, "Chapter 46: Simultaneous localization and mapping," Springer Handbook of Robotics, ISBN:978-3-319-32552-1, Springer Nature,1153–1176, 2016.

[3] S. Huang, G. Dissanayake. "Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM", *IEEE Transactions on robotics*, Vol(23), 1036-1049, 2007.

[4] S. Huang, G. Dissanayake, "Convergence analysis for extended Kalman filter based SLAM," *IEEE International Conference on Robotics and Automation, 2006*, Orlando, FL, USA, 2006.

[5] http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam05-ekf-slam-4.pdf

[6] Murphy, Kevin, and Stuart Russell. "Rao-Blackwellised particle filtering for dynamic Bayesian networks." *Sequential Monte Carlo methods in practice*. ISBN: 978-1-4757-3437-9, Springer, 499-515, 2001.

[7] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem", In AAAI National Conference on Artificial Intelligence, 2002.

[8] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, Vol(4), 23-33, 1997.

[9] R. Kümmerle, B. Steder, C. Dornhege, ark.*"*On measuring the accuracy of SLAM algorithms." *Auton Robot, Vol(*27), 387–407, 2009.

[10] https://cyberbotics.com/doc/guide/introduction-to-webots

[11] https://cyberbotics.com/doc/reference/index

[12] http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam12-fastslam-4.pdf