Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Virtual Fence Mapping Using Slam in Animal Barn Environments

Jung Kyu Park[1], Eun Young Park[2]

[1]Department of Computer Software Engineering, Changshin University, Changwon-si, Korea
[2]Rinascita College of Liberal Arts and Sciences, Shinhan University, Uijeongbu-si, Korea

Robots have been used for some time in the industry to perform certain tasks. However, these robots are static and not adaptable to the environment. In order for the robot to autonomously explore an unknown environment, it is necessary to know its location within the environment. This is called localization. The robot can only know where it is in the environment through a map. This results in requirements for mapping. Therefore, the robot must create a map and localize within it. This is called concurrent localization and mapping. In this paper, we use Adaptive Monte Carlo Localization (AMCL), an application of particle filters. A particle filter represents a belief with n samples or a set of particles, and each particle has a related weight. The built-in AMCL driver was implemented on the T815 robot and the results demonstrate that the robot can localize within a given environment. dards.

Keywords: AMCL, Mobile Robot, SLAM, Smart Farm, Virtual Fence

1. Introduction

These days, robots are becoming more practical in simple everyday functions. Robots have been in use for some time in industrial assembly lines to carry out certain tasks, for example arms and manipulators are used to assemble cars and in the processing of chemicals in labs etc. However, most of these robots are static and

do not adapt to the environment. The problem arises when a robot is used to navigate through an unfamiliar environment autonomously and carry out tasks such as vacuuming a certain area, moving objects from one place to another etc. For a robot to behave in this manner it needs to know its location in the environment, this is called localization. The robot can only know where its location in the environment via a map.

Therefore, the requirement for map building arises. Map building and localization are related tasks. As a robot can only determine its position with a map and with a reasonable estimate of its position, it is still being determined where to place the detected obstacles on the map. For this reason, a robot needs to build a map and localize it within this map at the same time. This is also known as Simultaneous Localization and Mapping [1-3].

The paper aims to integrate localization and mapping into the robot system using the existing software. However, it was a challenge to understand the concepts of SLAM and to implement it into the complex systems that were provided [4,5].

The following sections describe most common localization techniques used and particle filter mapping. Section 3 describes the design options, constraints, and project block diagrams after implementation. In Section 5, the obtained tests and results were shown. In Section 6, the conclusion was shown.

2. Literature Review

Robot localization being a hot research topic has gained the interests of many people and over the years a number of localization techniques have been invented. The following sections 2.1, 2.2 and 2.3 briefly explain some of the estimation algorithms used to localize a robot. The concept of an estimation algorithm was to express the robot's position as probability. Or "What is the probability that the robot is in this position?" The advantage of the estimation algorithm is that it is somewhat immune to measurement noise.

Bayesian filters

To localize a robot, it requires to gather the information of the environment using sensors. However, no sensors take perfect measurements or work well in all situations [6-8]. Due to the sensors not being perfect, the uncertainty can be represented with statistical tools such as Bayes filters.

The Bayes filter estimates the state from noisy observations [7-9]. The Bayes filter represents the state of time t with an arbitrary variable x_t, and the probability of state at each point in time is called belief, Bel(x_t) which also represents the uncertainty [6,9]. The Bayes filter continuously estimates this belief in the state domain for all the information.

$$Bel(x_t) = p(x_t|z_1, z_2, \dots, z_t) \quad (1)$$

Where z_1,z_2,…,z_t are the sensor observations. The belief, Bel(xt) represents this as what the probability that the robot is at location x_t if the history of sensor measurements is z_1,z_3,…,z_t is. However, the process gets more complicated and grows over time; this is because the sensor measurements increases [7].

Bayesian filter is also a recursive algorithm so it consists of an update step and a correction step. The following one-dimensional model illustrates these steps. In this graphical representation, the environment consists of three doors whose initial robot location is unknown. The robot is equipped with sensors that can detect but cannot distinguish between doors [6, 9]. Figure 1(a), the initial location of the robot is unknown, which is shown as a uniform distribution of possible locations. When the robot moves and reaches the first door, the sensor signals "Door Found". The trust function then places a high probability where the statement is and a low probability everywhere else. The distribution also shows three peaks, each of these peaks represents a door and these are the possible locations where a robot can be within this environment.

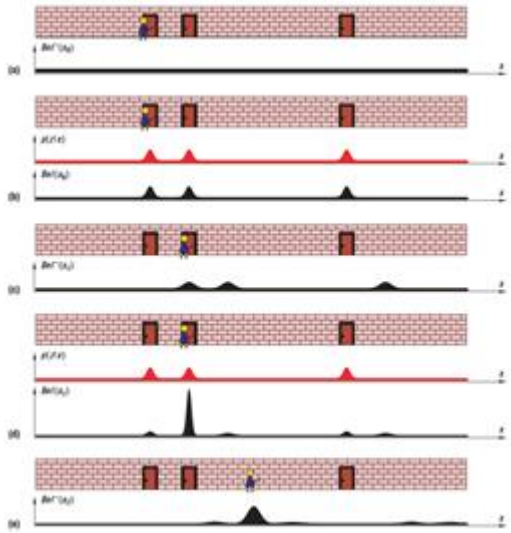


Fig. 1: One-dimensional Bayesian filter model [2]

Figure 1 also illustrates that the system can handle multiple hypothesis that conflict each other. As the robot moves to the right, the Bayes filter moves and smoothes the belief in the same direction as the robot's movement to explain the inherent uncertainty of the estimate (see Figure 1(c)). Finally, Figures 1(d) and 1(e) show the belief after the sensors observe the other door. When a robot moves, most probabilities are placed close to one of the doors, and the filter is fairly confident about the robot's position. The following equations show how the Bayes filter is updated, when a sensor provides new observations z_t (2) and the correction of the predicted estimate using the sensor observation (3).

$$Bel^{-}(x_t) \leftarrow \int p(x_t|x_{t-1})Bel(x_{t-1})dx_{t-1} \quad (2)$$

$$Bel(x_t) = \alpha_t p(z_t|x_t)Bel^{-}(x_t) \quad (3)$$

Where p(x_t |x_(t-1)) shows how the state changes over time and in equation 3, the perceptual model, p(z_t |x_t), describes "the likelihood of making observation z_t, given that the persons are at location x_t"[9]. A normalizing constant α_t ensures that the total posterior distribution sums up to one. The detail explanations can be found in [6,7].

Bayes filters are at an abstract level, meaning they only provide a probabilistic framework for recursive state estimation. A perceptual model, dynamics, and belief representation should be specified to implement a Bayes filter. The implementation of the Bayes filter differs by representing the probability density for the state x_t.

The common implementations of Bayes filters are the Kalman filters and the particle filters, which are briefly explained below.

Kalman filters

Kalman filters are optimal estimators [10-12]. Since most systems are not linear, Kalman filters have been extended by applying first order Taylor series and are known as Extended

Kalman filters (EKF) [11]. The advantaged of using Kalman filters are their computational efficiency and accuracy when are given good inputs [12]. The calculation process is much faster, but the Kalman filter can represent only a single-modal distribution, so the cost of expression is high [6]. The accuracy can be improved by having accurate sensors and by providing the initial position of the robot. Kalman filters are also used as a bench mark to compare against other algorithms [11, 12].

Particle filters

A particle filter represents a belief with n samples or a set of particles [13-15].

$Bel(x_t) \approx S_t = \{\langle x_t^{(i)}, w_t^{(i)} \rangle i=1,\ldots,n\}$ (4)

$x_t^{(i)}$: represents each state,

$w_t^{(i)}$: non-negative important factors, which sum up to 1.

The Figure 2 illustrates the same one-dimensional model from the Bayesian filters but now implemented by using particle filters. The initial position of the robot is unknown so a uniformly distrusted sample set represents the robot's position [6]. Therefore, the initial weights of the particles are of the same importance as it can be seen from Figure 2a. As the robot moves to the right (Figure 2(b)), the sensor will detect the first door. And the particle filter adjusts and normalizes each critical factor in the sample to incorporate measurements [2]. The sample (Figure 2(b)) is the same as before, but now there are important factors proportional to the observability.

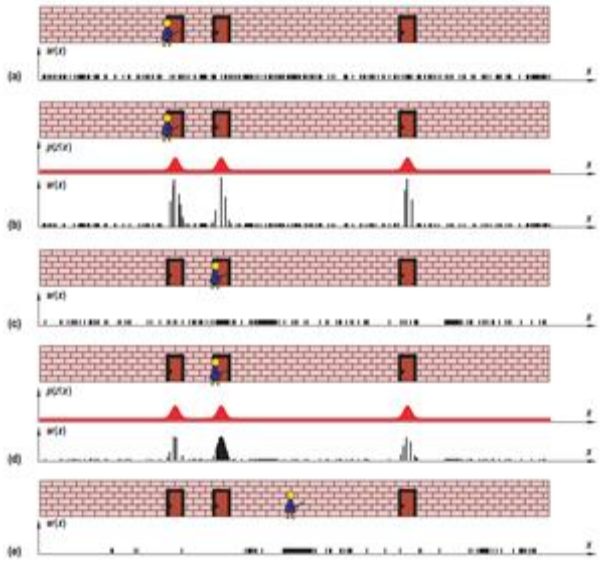


Fig. 2: One-dimensional Particle filter model

Filters estimate possible positions for each new particle. This is similar to the Bayesian prediction step (Equation (2)). As a result, the resulting set of samples may differ from the original, with most of samples concentrated in three locations. In Figure 1(d), the sensor detects the second door, leading to a possible p(z|x). The sample set (shown below in Figure

1(d)) can be obtained by weighting the importance factor proportional to the likelihood probability. After the next prediction step, most of the particles converge and concentrate on the robot's true position.

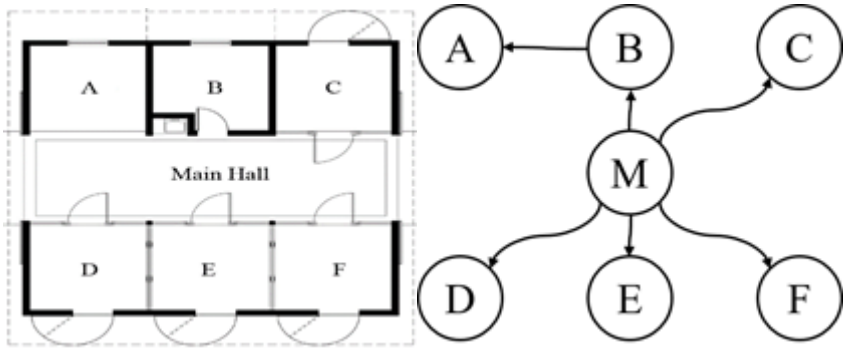
The advantage of particle filters is that they can converge to the real rear even in non-Gaussian nonlinear dynamic systems [13, 14]. Particle filters are very efficient because they focus only on areas of high probability state space. Several improvements have been made to particle filters to make available samples more efficient. An example of this is Adaptive Monte Carlo Localization (AMCL) [15,16].

AMCL increases the efficiency of particle filters by immediately resizing the sample set [16]. The name KLD is because the approximation of the error is measured by the Coolback-Ribler distance. More information about this technology can be found in [9].

The AMCL technique has been implemented in player modules by Professor Andrew Howard from the University of Southern California and this driver can be used to localize a robot within a given environment but is dependent on a predefined map.

Map representation and mapping

Maps can be classified as either metric or topological. Figure 3 shows example of metric and topological map. A topological map shows the connectivity between significant places or features whereas a metric map records the distance between mapped entities. The metric map is often taken as a 2D projection of a 3D environment, much like a bird's eye view. The following figures are examples of topological and metric maps. In a metric map the measured distance is proportional to the physical separation. These types of maps are accurate but require more resources to store. A low-resolution metric map is common in robot navigation [12].



(a) Metric map (b) Topological map

Fig.3: Mapping results using pmap

3. Experiment Design

The main requirement is to implement localization and mapping to gather information from the surrounding environment. Therefore, to localize the robot within the environment,

exogenous sensors such as infrared, laser, and sonar must be installed.

3.1. PLAYER/STAGE

The player's client/server model allows the robot to be programmed in different languages. Client programs can run on any computer that can connect to the robot [17].

Stage is a multi-robot simulator. Simulate the number of sensors, mobile robots, and objects in a 2D environment. It is designed for research and debugging purposes. The client programs can be run on stage instead of the real robot. The outcomes of the client programs would be similar if it were to run on a real robot. Figure 4 illustrates the structure of Player/Stage simulation software.

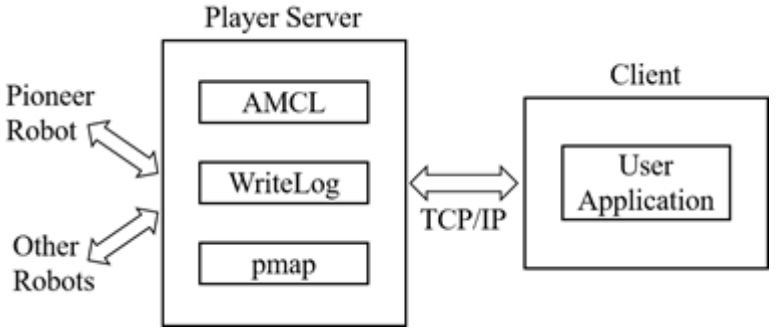


Fig. 4: Block diagram of Player/Stage

4. Research Methodology

Having a number of robots to choose from and the required software being available, it was decided to first implement localization on the mobile robot. This is because the mobile robot was not very complex to understand as it was controlled by two microprocessors.

4.1. IMPLEMENTING THE MOBILE ROBOT

The first tasks that needed to be accomplished before the implementation of localization on the mobile robot were to achieve wireless communication and integration with player/stage modules. Block diagram of Figure 4 shows the connections between the mobile hardware and the software modules

To achieve wireless communication the robot's existing code had to be modified. This is because the existing code was only compatible with the old mobile robot which had a sonar sensor, an infrared sensor and a single microprocessor. However, the new mobile robot has two ATMEL microprocessors (mega 8 and mega 32). An mobile phone was used as a wireless link; it communicates with the mobile robot via a serial port and the software running on the mobile phone simply forwards the characters from the serial port to the network and vice-versa.

Localization was not implemented on mobile robot as it was unfeasible to use one sonar sensor to gather the information of the surrounding environment. On the other hand, there were other options such as the b21r, which is equipped with many sonar sensors and also a laser range

finder which is superior compared to sonar as it has 181 degrees of vision. Therefore, it was decided to implement localization on the TT8 robot.

Nevertheless, the first steps have been taken which contribute towards localization by achieving wireless communication and the integration with player modules and this can be further developed to integrate localization on the mobile robot.

4.2. LOCALIZATION

Figure 5 shows the TT8 mobile Robot that includes the embedded board. To implement localization on the TT8 robot it was decided to utilize the built-in AMCL driver of the player. However, the following things needed to be carried out in order for the robot to localize.

Fig. 5: TT8 mobile robot with embedded board

The player server is reconfigured so that the AMCL driver is enabled. This is very crucial as the client files which are the application files need this driver to implement localization. The player is also configured with the rtkgui. This is the GUI which is used for debugging purposes and to display the particle distribution and the location of the robot on the map as time progresses.

Configuration files are files that are used to configure the player. All the devices are instantiated within configuration file. A Player/Stage configuration file that was written to implement localization. The file basically instructs the server to support the devices declared in it. Client files (application files) can connect to these devices and will be able to read the data and give commands.

A number of things in the mapfile driver, AMCL driver and writelog driver can be intialized. For example, in the mapfile driver the path to the predefined map and the resolution of the map and in the AMCL driver the maximum number of samples to use and the initial position of the robot can be initialized.

Metric maps of the environment are created as the AMCL algorithm require a predefined map. Therefore, detailed metric maps were developed by creating mazes and measuring the distances between corridors and walls. Refer to Figure 8 for an example of a metric map. The maps created should represent the environment accurately as the algorithm depends on it. If

the maps were incorrect the robot will not localize properly. The resolution of the map was of high importance as well, as the map will be represented as a ratio of pixels to centimeters.

4.3. MAPPING

The following things were required to be accomplished in order to implement mapping on TT8 robot. Modifying player writelog library The player writelog library is modified so that the correct data is only written to a log file. The data which requires to be written by the writelog file are the position data and the laser data and also a special END message is required to terminate the mapping process.

The Pmap library is also modified so that the mapping process is online. This required modifications to the way the pmap reads the log file. The log file is continuously read and the map is updated every 10 seconds.

5. Result and Discussion

Tests on localization and mapping were conducted by building mazes and also by initializing the robot's initial position on the map.

5.1. TESTING ODOMETRY AND LOCALIZATION

Test was carried out by creating a rectangular environment of 4x5 meters. The initial position is set as (0, 0, 0). The robot was programmed to go in a loop and stop once it gets back to the initial position, this was repeated several times. This same test was then carried out by using localization.

The test proves that accurate localization cannot be achieved by using pure odometry as error accumulates over time as the robot moves. Figure 6 shows the difference between the localization and the odometry.

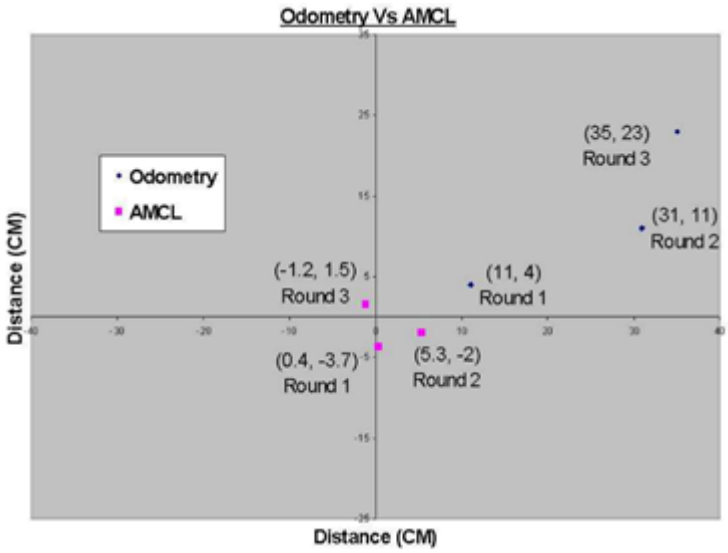


Fig. 6: Comparison between odometry and AMCL

5.2. TESTING LOCALIZATION WITH KNOWN ROBOT POSITION

The test was performed by initializing the robot's position in the client file. The robot's initialized position on the predefined map was almost identical to the robot's position in the real world. Therefore, it is a "known location" situation. The following screenshot shows you using a laser to locate the robot as it moves around its surroundings to avoid obstacles.

As shown in [Figure 7], before the robot starts moving, the map is filled with particles (yellow circles around the particles) and each particle is weighted. Each particle represents the robot's position (x, y, angle). Particles begin to converge as the robot moves through its surroundings. See Figure 8. This means that the number of places where the robot can be placed decreases, as shown in [Figure 9], and eventually the robot is localized.

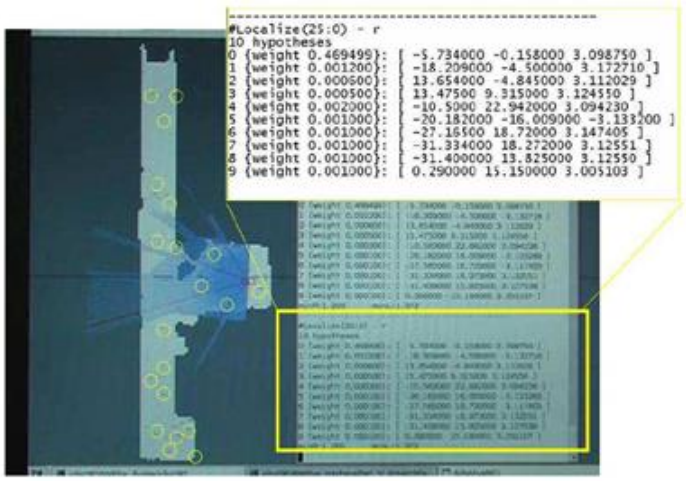


Fig. 7: Map filled with particles and each particle having a weight

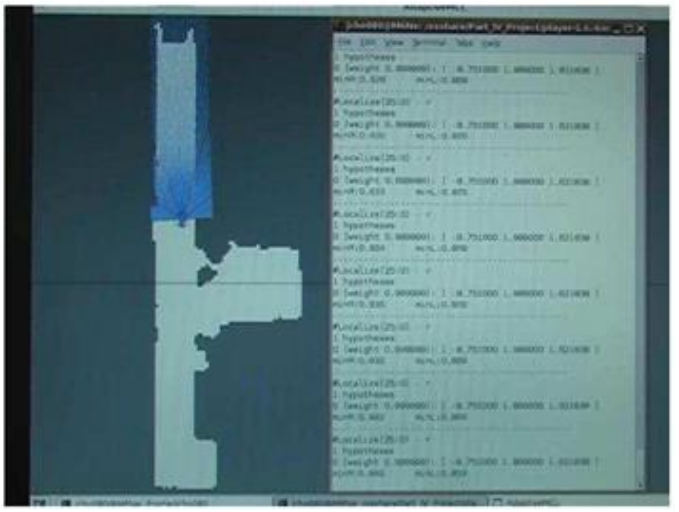


Fig. 8: Particles converging

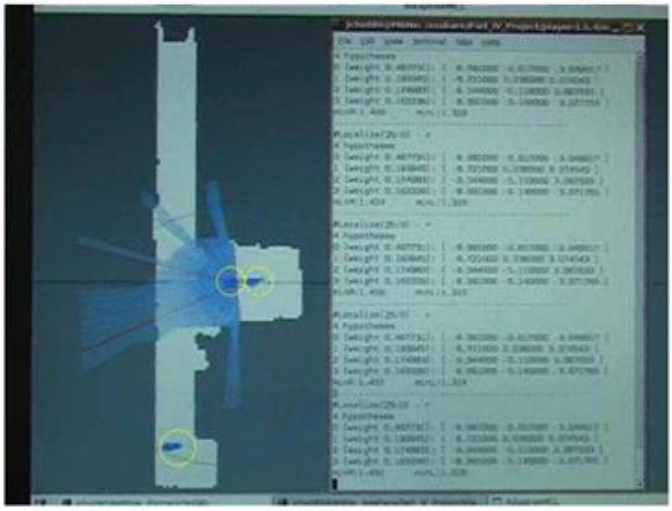


Fig. 9: Particles converged to one location and the robot

However, the above test was repeated again by initializing the robot position to unknown. The robot's position on the predefined map is different from the actual robot's position. Figure 10 shows the initial position given and the actual position of the robot. A red circle represents the actual position of the robot in the real world, and a yellow circle represents the initial position of the robot specified on a predefined map.

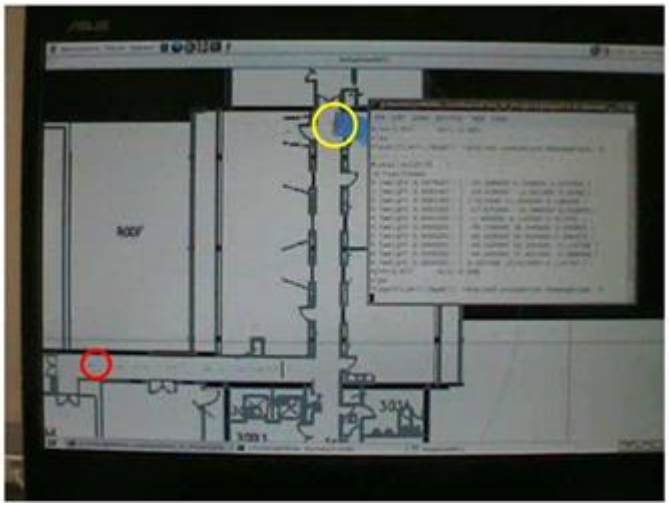
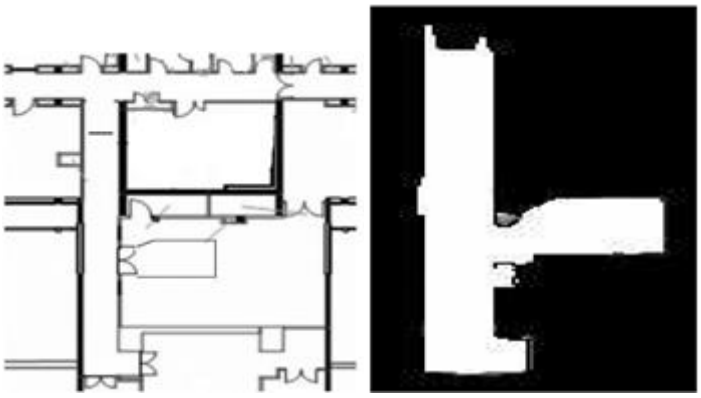


Fig. 10: Robots true position (red circle) and the given position (yellow circle)

5.3. TESTING ON MAPPING

Figure 11 shows the result maps. Mapping tests were conducted by using the pmap library, the following figures show very little differences between the floor plan and the map drawn by the robot using pmap library.



(a) Floor plan of barn (b) Map drawn by using pmap

Fig. 11: Mapping results using *pmap*

6. Conclusion and Future Work

The paper goal of integrating localization and mapping (SLAM) into the robot system was successful. The robot is able to localize and map at the same time, however it requires a predefined map. There were several issues encountered with the localization of the robot. The robot would not localize properly if the maximum number of particles were set very low such as 100 particles. This is because the number of places where a robot can be located has decreased and the particles converge rapidly due to the small set size of the particles. Increasing the size can solve the problem but comes at a computational cost. The maximum samples used in the tests were 10,000 samples, with this the robot was able to localize from an unknown position.

Future modifications to the integration of SLAM into the robot system would be to modify the AMCL algorithm so that it does require a predefined map. A possible solution would be to use pmap to map a certain section of the area and allow the robot to localize within this section and repeat these steps until the robot has covered the entire environment.

ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2021R1F1A1052129)

References

1. Durrant-whyte, H. & Bailey, T. (2006) Simultaneous Localisation and Mapping: Part I, IEEE Robotics & Automation Magazine, 13(2), 99-108. https://doi.org/10.1109/MRA.2006.1638022
2. Bailey, T. & Durrant-whyte, H. (2006) Simultaneous Localisation and Mapping: Part II, IEEE Robotics & Automation Magazine, 13(3), 108-117. https://doi.org/10.1109/MRA.2006.1678144

3. Lajoie, P.Y., Ramtoula, B., Chang, Y., Carlone, L. & Beltrame, G. (2020) DOOR-SLAM: Distributed, Online, and Outlier Resilient SLAM for Robotic Teams, IEEE Robotics and Automation Letters, 5(2), 1656-1663. https:// 10.1109/LRA.2020.2967681.
4. Girerd, C., Kudryavtsev, A. V., Rougeot, P., Renaud, P., Rabenorosoa, K. & Tamadazte, B. (2022) Automatic Tip-Steering of Concentric Tube Robots in the Trachea Based on Visual SLAM, IEEE Transactions on Medical Robotics and Bionics, 2(4), 582-585. https://10.1109/TMRB.2020.3034720.
5. Zhou, H., Yao, Z., Zhang, Z., Liu, P. & Lu, M. (2022) An Online Multi-Robot SLAM System Based on Lidar/UWB Fusion, IEEE Sensors Journal, 22(3), 2530-2542. https://10.1109/JSEN.2021.3136929.
6. Fox,V., Hightower, J., Liao, L., Schulz, D. & Borriello, G. (2003) Bayesian filtering for location estimation, IEEE Pervasive Computing, 2(3), 24-33. https:// 10.1109/MPRV.2003.1228524.
7. Blanco, J. L., FernÁndez-Madrigal J. A., & GonzÁlez, J. (2008) Toward a Unified Bayesian Approach to Hybrid Metric--Topological SLAM, IEEE Transactions on Robotics, 24(2), 259-270. https://doi: 10.1109/TRO.2008.918049.
8. Krishnamurthy, V. (2020) Convex Stochastic Dominance in Bayesian Localization, Filtering, and Controlled Sensing POMDPs, IEEE Transactions on Information Theory, 66(5), 3187-3201. https://10.1109/TIT.2019.2948598.
9. Thrun, S., Burgard, W. & Fox, D. (2005) Probabilistic Robotics, The MIT Press.
10. Chen, S. Y. (2012) Kalman Filter for Robot Vision: A Survey, IEEE Transactions on Industrial Electronics, 59(11), 4409-4420, https://10.1109/TIE.2011.2162714.
11. Ullah, I., Shen, Y., Su, X., Esposito, C. & Choi, C. (2020) A Localization Based on Unscented Kalman Filter and Particle Filter Localization Algorithms, IEEE Access, 8, 2233-2246. https://10.1109/ACCESS.2019.2961740.
12. Fang, Y., Panah, A., Masoudi, J., Barzegar, B. & Fatehi, S. (2022) Adaptive Unscented Kalman Filter for Robot Navigation Problem (Adaptive Unscented Kalman Filter Using Incorporating Intuitionistic Fuzzy Logic for Concurrent Localization and Mapping), IEEE Access, 10, 101869-101879. https://doi: 10.1109/ACCESS.2022.3207925.
13. Luo, J. & Qin, S. (2018) A Fast Algorithm of Simultaneous Localization and Mapping for Mobile Robot Based on Ball Particle Filter, IEEE Access, 6, 20412-20429. https://10.1109/ACCESS.2018.2819419.
14. Zhang, Q., Li, Y., Ma, T., Cong, Z. & Zhang, W. (2021) Bathymetric Particle Filter SLAM With Graph-Based Trajectory Update Method, IEEE Access, 9, 85464-85475, https://10.1109/ACCESS.2021.3088541.
15. Xiong, J., Cheong, J.W., Ding, Y., Xiong, Z. & Dempster, A. G.(2022) Efficient Distributed Particle Filter for Robust Range-Only SLAM, IEEE Internet of Things Journal, 9(21), 21932-21945. https://10.1109/JIOT.2022.3181994.
16. Chung, M. A., & Lin C. W. (2022) An Improved Localization of Mobile Robotic System Based on AMCL Algorithm," IEEE Sensors Journal, 22(1), 900-908, https://10.1109/JSEN.2021.3126605.
17. Xu S. & Chou W. (2017) An Improved Indoor Localization Method for Mobile Robot Based on WiFi Fingerprint and AMCL, 2017 10th International Symposium on Computational Intelligence and Design (ISCID). 324-329, https://10.1109/ISCID.2017.25.
18. The Player/Stage Project, http://playerstage.sourceforge.net
19. Prieto, G. A. & Mendoza, J. P. (2012) Low Cost Didactic Robotic Platform Based on Player/Stage Software Architecture and La Fonera Hardware, IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 8(3), 126-132. https:// 10.1109/RITA.2013.2273112.
20. Cho, O.H. (2024). An Evaluation of Various Machine Learning Approaches for Detecting Leaf Diseases in Agriculture. Legume Research. https://doi.org/10.18805/LRF-787

21. Maltare, N. N., Sharma, D., Patel, S. (2023). An Exploration and Prediction of Rainfall and Groundwater Level for the District of Banaskantha, Gujrat, India. International Journal of Environmental Sciences, 9 (1), 1-17. https://www.theaspd.com/resources/v9-1-1-Nilesh%20N.%20Maltare.pdf
22. Min, P.K., Mito, K. and Kim, T.H. (2024). The Evolving Landscape of Artificial Intelligence Applications in Animal Health. Indian Journal of Animal Research. https://doi.org/10.18805/IJAR.BF-1742