Jurnal
Teknologi

Full Paper

PARTICLE FILTER IN SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM) USING DIFFERENTIAL DRIVE MOBILE ROBOT

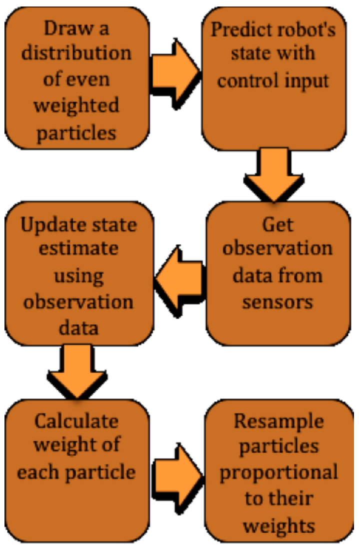
Norhidayah Mohamad Yatim[a,b], Norlida Buniyamin[a]

[a]Faculty of Engineering, Universiti Teknologi MARA (UiTM), Shah Alam, Selangor, Malaysia
[b]Faculty Electronics and Computer Engineering, Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia

*Corresponding author
norhidayahm@
utem.edu.my

Article history
Received
15 May 2015
Received in revised form
1 July 2015
Accepted
11 August 2015

Graphical abstract


Draw a distribution of even weighted particles
Predict robot's state with control input
Get observation data from sensors
Update state estimate using observation data
Calculate weight of each particle
Resample particles proportional to their weights

Abstract

Simultaneous Localization and Mapping (SLAM) problem is a well-known problem in robotics, where a robot has to localize itself and map its environment simultaneously. Particle filter (PF) is one of the most adapted estimation algorithms for SLAM apart from Kalman filter (KF) and Extended Kalman Filter (EKF). In this work, particle filter algorithm has been successfully implemented using a simple differential drive mobile robot called e-puck. The performance of the algorithm implemented is analyzed via varied number of particles. From simulation, accuracy of the resulting maps differed according to the number of particles used. The Root Mean Squared Error (RMSE) of a larger number of particles is smaller compared to a lower number of particles after a period of time.

Keywords: SLAM, mapping; particle filter, e-puck

1.0 INTRODUCTION

Many notable researches have been carried out in attempts to solve both localization and mapping problem in unknown environment of robotic system. This problem is famously known as the Simultaneous Localization and Mapping (SLAM) problem. A SLAM algorithm is about exploring the environment to build its map and determining the location of the robot in the map. There are many application of SLAM solution, such as in forest harvesting, minimal invasive surgery and autonomous vehicles.

Two obvious fundamental tasks consist in SLAM are localization and mapping. Localization means to determine robot's position and orientation while mapping is to map the environment.

The essential objective of solving SLAM is for a robot to map an unknown environment while exploring the environment. The map obtained can be used for countless possibilities.

In an outdoor environment a robot could obtain its current position using Global Positioning System (GPS) signal. However, GPS has its own limitation where signals can be shadowed by buildings, terrains or even when it is raining. This condition is called GPS blind area. Thus, there are circumstances where GPS signal might not allow the robot to locate itself in the environment.

Solving the SLAM problem might not be necessary if either one of the two information (i.e. map of the environment or robot's location) is available to the robot. SLAM topic has been extensively studied in the literature. The implementation of SLAM solution in real life situation as well has begun to show successes in applications.

This paper aims to provide an overview of researches that has been carried out in implementation of particle filter in SLAM problem. Section 2 covers the previous works in particle filter SLAM. Section 3 and 4 describe the estimation theory of Bayes filter and its implementation in this work. Section 5 shows the result and analysis that has been obtained using particle filter SLAM. The last section concludes the simulation result, challenges and future works.

2.0 PREVIOUS WORK

There has been significant progress in SLAM algorithm applied in indoor [1]–[5] or outdoor environment because the topic has gained much attention by researchers. However, there are still challenges that need to be encountered [6]. The most popular estimation approaches in SLAM algorithm in the literature are Extended Kalman Filter (EKF) and Particle Filter. In EKF SLAM algorithm two main assumptions were made; 1) Features or object in the environment could be uniquely associated with the sensor measurements also known as data association problem and 2) Noise in sensor's measurements and robot's trajectory follow Gaussian distribution.

The advantage of particle filter over EKF is instead of model linearization, it uses some samples to get the state estimation. Unlike EKF, particle filter can process raw data from sensor without the need of landmark or feature detection. Particle filter was first applied in SLAM problem by Murphy, Doucet, and colleagues [7], [8]. Using the particle filter, each particle in the sample represents the robot trajectory and a map. This however increases memory usage and computation cost.

Rao-Blackwellized particle filter (RBPF), derived from particle filter, was then applied in an algorithm proposed by Montemerlo named FastSLAM [9]. FastSLAM compensate the amount of memory usage by sharing map between particles, but requires predetermined landmark in the environment. FastSLAM adopted landmark-based map and uses laser range scan as robot sensor. The first version of the algorithm; FastSLAM 1.0 utilizes EKF to keep landmark based map of each particle. The second version; FastSLAM 2.0 uses EKF to generate a better proposal distribution of the particle filter [10].

FastSLAM algorithm using occupancy grid map representation instead of feature-based map was implemented in [11]. It improves the motion model adopted and reduced the number of particle for state estimation.

A compact and efficient map representation called Distributed Particle (DP) map was developed without the need of predetermined landmark in [12]. But the drawback was the use of complex tree data structure that increased the computation complexity. DP-SLAM used laser range scan as well and build an occupancy grid map, however it is inaccurate for very small objects. This approach was then improved by using adaptive proposal distribution instead of fixed proposal distribution in particle resampling [13]. Instead of resampling at a fix time duration, the resampling is determined adaptively. This could prevent particles from the degradation or what is called the particle depletion problem. The method was validated in large scaled indoor and outdoor environment without predetermined landmark. Another method in [14] as well managed to build an accurate grid map without predetermined landmark. However, the focus was on closing the loop in the environment especially in nested loops. A fast computation speed RBPF based algorithm was then developed in [15] by utilizing previously computed distribution proposal and a compact form of map.

3.0 ESTIMATION THEORY: BAYES FILTER IN SLAM

In SLAM problem, as explained previously, map of robot's environment needs to be developed based on robot's control input and sensors readings. Since both data contains uncertainties from noises, it is necessary to apply an estimation theory to develop a map from this noisy information.

In robotics, Bayes rule is a powerful tool to perform estimation. Bayes rule states that a posterior probability of an event A given an event B is equal to the likelihood of B given A, multiplied with α, a normalization factor as shown in equation 1.

$$P(A|B) = \alpha P(B|A)P(A) \tag{1}$$

Bayes rule is applied to SLAM problem where a robot will estimate its position in order to obtain a map of its environment over time. Thus, to improve the accuracy of its map, the robot needs to have a good estimate of its position also known as state of the robot pose. This is denoted using x_k, which is the state of robot pose at time k. To model the uncertainties, the state estimate, x_k is maintained in a probability distribution function (pdf) instead of a single value. The pdf of state estimate can be obtained conditioned on robot's observation, z_k. A general Bayes rule for this probability is stated in equation 2. This term is called a posterior.

$$p(x_k|z_k) = \alpha p(z_k|x_k) \times prior \tag{2}$$

$$prior = px_k|x_{1:k-1}, z_{1:k-1}, a_{1:k} \tag{3}$$

Here, $prior$ denotes our best guess or prediction's distribution of the state before we integrate robot's observation. Initially we include all previous states, observations and previous control inputs to obtain the distribution of prior. Equation 3 is a general Bayes rule for prior where $a_{1:k}$ is control inputs up to time k.

A common notation for posterior distribution in equation 2 is $bel(x_k)$, which implies the robot's belief of current state's estimate. While $\overline{bel}(x_k)$ is a notation for $prior$ distribution, where the bar indicates that robot's belief is still a prediction of the state. Notice that in equation 4 the latest observation, z_k is not yet incorporated.

$$\begin{aligned} bel(x_k) &= \alpha p(z_k|x_k) \times \overline{bel}(x_k) \\ \overline{bel}(x_k) &= p(x_k|x_{k-1}, z_{1:k-1}, a_{1:k}) \end{aligned} \tag{4}$$

The Bayesian network diagram in Figure 1 illustrates robot's movement from time $k-1$ to $k+1$. Each move is caused by action control a and resulting an observation z. It is shown that the prior, $\overline{bel}(x_k)$ is obtained before incorporating current observation, z_k. While the posterior, $bel(x_k)$, is obtained after that. It is also noted that current state x_k only depends on previous state, x_{k-1}, rather than all previous state, $x_{1:k-1}$. This simplifies the $prior$ in 3.

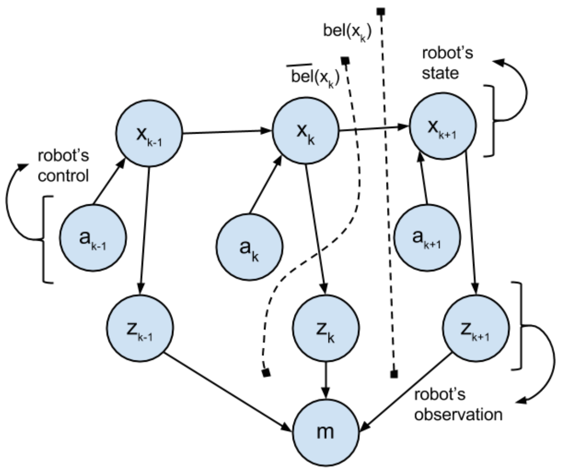


Figure 1 Bayesian Network describes robot movement at time $k-1$ to $k+1$. Here, robot's pose denotes by x is applied with a control, a to make the next movement. At each time step, it makes an observation, z and these observations form a map of the environment, m

From the general Bayes rule in equations 2 and 3, we want to write $bel(x_k)$ as a function of previous state distribution, x_{k-1}.

Referring back to Bayes rule in equation 1, we can obtain the distribution of $P(B)$ by using the Law of Total Probability in equation 5. Here, we calculate the convolution of pdfs, which sums up all the possibilities of A instances.

$$P(B) = \sum_{i=1} P(B|A_i) \times P(A_i) \tag{5}$$

By applying equation 5 to the $prior$, $\overline{bel}(x_k)$, a discrete probability distribution for $\overline{bel}(x_k)$ can be obtained as equation 6.

$$\overline{bel}(x_k) = \sum_{x_{k-1}} p(x_k|x_{k-1}, z_{1:k-1}, a_{1:k}) \times p(x_{k-1}|z_{1:k-1}, a_{1:k}) \tag{6}$$

However, since our pdf is a continuous distribution, the sum is changed into integral as in equation 7.

$$\overline{bel}(x_k) = \int p(x_k|x_{k-1}, z_{1:k-1}, a_{1:k}) \times p(x_{k-1}|z_{1:k-1}, a_{1:k}) dx_{k-1} \tag{7}$$

To simplify equation 7, again refer to the Bayesian network in Figure 1. This diagram illustrates that if we know the previous state of robot pose, x_{k-1} and current control a_k, we do not need all previous observation, $z_{1:k-1}$ and all previous control, $a_{1:k-1}$, to determine the distribution of current state x_k. Thus, the term can be simplified as equation 8.

$$p(x_k|x_{k-1}, z_{1:k-1}, a_{1:k}) = p(x_k|x_{k-1,} a_k) \tag{8}$$

For the second term in 7, using the same diagram, it is observed that previous state, x_{k-1} does not depend on current control a_k. So this term can be taken out which leave the second term as follows:

$$p(x_{k-1}|z_{1:k-1}, a_{1:k}) = p(x_{k-1}|z_{1:k-1,} a_{1:k-1}) \tag{9}$$

Now, equation 9 can be rewritten as the belief of state estimate at time $k-1$, $bel(x_{k-1})$

$$p(x_{k-1}|z_{1:k-1,} a_{1:k-1}) = bel(x_{k-1}) \tag{10}$$

If we rewrite term 8 and 10 into the general Bayes rule in 2 and 3, we obtain the algorithm of Bayes filter in recursive manner as written in Algorithm 1.

```
Algorithm 1 Bayes Filter Algorithm
Require: $bel(x_{t-1}), z_t, a_t$
  $\overline{bel}(x_t) = \int p(x_t|x_{t-1}, a_t) \times bel(x_{t-1}) dx_{t-1}$
  {Prediction Step}
  $bel(x_t) = \alpha p(z_t|x_t) \times \overline{bel}(x_t)$ {Correction Step}
  return $bel(x_t)$
```

Bayes filter is a powerful tool in robot estimation where a robot's state is estimated over time using previous state. As stated in Algorithm 1, $\overline{bel}(x_k)$ is the prediction step, where we predict what is the current state base on previous state estimate and current control input. While $bel(x_k)$ is the correction step, where we incorporate the latest observation into the estimate of current state.

In SLAM problem, Bayes filter requires a motion model to obtain the distribution $p(x_k|x_{k-1,} a_k)$ and an observation model that describes the robot's observation and obtain the distribution $p(z_k|x_k)$. However, it is known that to solve Bayes filter in closed form, the motion model and observation model has to be a linear system and noise in the system should follow Gaussian distribution. But if the motion model and observation model are non-linear systems, and the noise is not Gaussian, it will be harder to solve the Bayes filter.

In particle filter, instead of using probability distribution, we use approximation method to describe a distribution. Section 4 will introduce particle filter algorithm in SLAM problem.

4.0 PARTICLE FILTER

Particle filter is a method where a set of particles is used to represent robot's belief, $bel(x_k)$ instead of using parametric values (i.e. mean, μ and variance, σ^2) to describe $bel(x_k)$ distribution such as in Kalman Filter and Extended Kalman Filter. Figure 2 shows a set of particles in circles represent the belief of a robot pose, x_k.

In particle filter, each particle contains a hypothesis of robot pose that assume its position is correct. Then, by using this assumption, every particle will maintain its own map. Figure 2 illustrates that a map plotted by the particles will be different. Some particles will have better accuracy than the others.

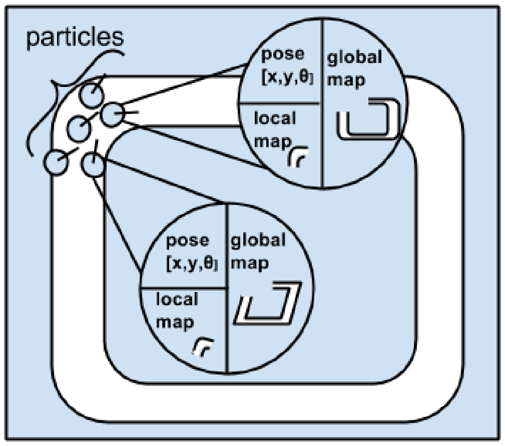


Figure 2 A set of particles represent a belief of robot pose marked by the circle. Each particle has its own map whereas some will be more accurate than others

There are several steps to implement particle filter algorithm. Below are the steps describing the algorithm:

- Draw a distribution of even weighted particles.
- Update robot's state in each particle with control command.
- Get observation data.
- Update the robot's state estimate by incorporating robot's observation.
- Calculate the importance weight of each particle using the difference between actual observation and predicted observation.
- Resample particles proportional to their weights.

These steps are put into the prediction step and correction step of the Bayes filter in order to implement the particle filter.

4.1 Prediction Step

The goal is to solve Bayesian filter using particle filter algorithm. In the prediction step, a set of particles is drawn from a distribution describing the robot pose, $x_k \sim p(x_k|x_{k-1}, a_k)$. To do this, we need a motion model to describe robot movement at every time step.

A motion model is a probabilistic way to estimate the robots state by using previous state and integrating the control and error a it is noted that to have a perfect reading over the robot motion and sensors measurements is naive and inadequate approach [16].

There are two types of motion model that can be adopted; Odometry-based model and velocity-based model. For this research, odometry-based model is used by utilizing the wheel encoders in the robot system. A motion model from section 5.2.4 of [17] is adapted.

Equation 11 describes the transition from previous state, x, y, θ to current state, x', y', θ' as a function of $g(x, y, \theta, l, r)$. Here, l and r are the traveled distance by left wheel and second wheel respectively, while b is the distance between two wheels (see Figure 3).

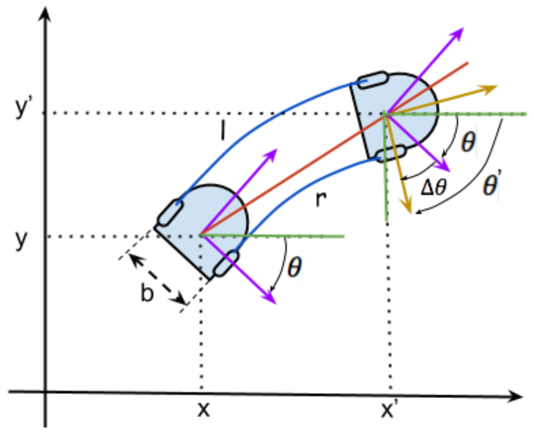


Figure 3 Robot moves from position $[x, y, \theta]$ to $[x', y', \theta']$. $\Delta\theta$ is the change of robot orientation where the calculation is $\Delta\theta = \frac{r-1}{b}$

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{r+l}{2}\cos\left(\theta + \frac{r-l}{2b}\right) \\ \frac{r+l}{2}\sin\left(\theta + \frac{r-l}{2b}\right) \\ \frac{r-l}{b} \end{bmatrix} \qquad (11)$$

Due to the noise from the wheel encoders, we need to model the resulting uncertainty. Thus, to estimate the robot state for every robot transition l and r are sampled from normal distribution $\mathcal{N}(l_t, \sigma_{l_t}^2)$ and $\mathcal{N}(r_t, \sigma_{r_t}^2)$. $\sigma_{l_t}^2$ and $\sigma_{r_t}^2$ are the variances from noise of the wheel encoder. These values can be obtain from equation 12 where α_1 and α_2 are translation error and rotation error from the wheel encoder respectively. The overall sampling step is described in algorithm 2.

$$\sigma_{l_t}^2 = (\alpha_1 \cdot l)^2 + \left(\alpha_2 \cdot (l-r)\right)^2$$
$$\sigma_{r_t}^2 = (\alpha_1 \cdot r)^2 + \left(\alpha_2 \cdot (l-r)\right)^2 \quad (12)$$

Algorithm 2 Sampling in Particle Filter Algorithm

```
For all $i = 1\ to\ M$ do
    $sample\ l_t' \sim \mathcal{N}(l_t, \sigma_{l_t}^2)$
    $sample\ r_t' \sim \mathcal{N}(r_t, \sigma_{r_t}^2)$
    $x_t^i = g\left(x_{t-1}^i, \begin{bmatrix} l_t' \\ r_t' \end{bmatrix}\right)$
end for
return $\{x_t^1, x_t^2, \ldots\ldots, x_t^M\}$
```

4.2 Correction Step

In the correction step, we incorporated robot's observation. In this work, occupancy grid map algorithm and map matching method are implemented. In occupancy grid map, the environment is made of number of grids with equal size. Each grid is called a cell. The probability of each cell is occupied is computed and represented using gray value. For this reason, it is also named as occupancy grid map in the literature. The grid-based map algorithm was first developed by Moravec and Elfes in the 80's [18]. The algorithm uses inverse sensor model, which is initially developed for noisy sonar sensor. The occupancy value of each cell depends on the measured distance, where the coincident cells are considered occupied (i.e. black cells) and the cells in between are considered free (i.e. white cells) as illustrated in Figure 4.

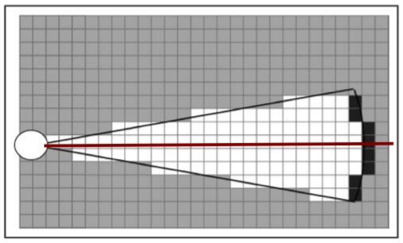


Figure 4 Cells that reside in the range of sensor measurements are marked as occupied (i.e. black cells) and the cells in between are marked as free (i.e. white cells) [19]

Equation 13 calculates the occupancy of each cell, m_i, as a sum of the log odd value of the cell given current observation, z_k, and current robot pose, x_k, and its recursive term. The last term is the initial value of the cell. The computation is efficient because the equation is a summation.

$$l(m_i|z_{1:k}, x_{1:k}) = l(m_i|z_k, x_k) + l(m_i|z_{1:k-1}, x_{1:k-1}) - l(m_i) \quad (13)$$

In map matching method, there is no need for probability distribution. However a local map that contains current measurement is compared with a global map that has past measurements. This requires each particle maintains a local map and a global map.

The weight of particle i at time k for each particle is calculated using a matching function in equation 14 adapted from [20]. Where $match$ is a number obtain from comparing local map and global map of a particle i. A larger match value indicates that the particular particle carries a more accurate map. f is a parameter that reflect the distribution of the particles' weight. In this experiment we set $f = 100$.

$$w_k^i = w_{k-1}^i \cdot e^{\frac{match^i}{f}} \quad (14)$$

After the particles are assigned with weights, the particles will be resampled proportionally to their weights. In this work, a resampling method named low-variance resampling is implemented. This method is adapted from Probabilistic Robotics by S. Thrun [21]. The next section presents the results that were obtained by implementing the particle filter algorithm described in this section.

5.0 RESULTS AND ANALYSIS

We implemented particle filter algorithm described in Section 4 in a simulation environment as shown in Figure 5, which consists of four objects. The mobile robot used, named, e-puck is a differential drive mobile robot with two motors. The e-puck robot is equipped with eight infrared proximity sensors surrounding the robot's edge.

Three set of particles were implemented; 100 particles, 300 particles and 500 particles. The effect of number of particles to the resulting map is later compared. The Root Mean Squared Error (RMSE) of each set of particles is calculated using equation 15. Here, N is the number of particles used. Since we are using robot simulator, the ground truth of robot's state can be obtain from the robot simulator, which is denoted by x_k. The estimated robot's state (i.e. from the particle filter algorithm) is denoted by $\hat{x}_k^i$ where i represents the ith particle.

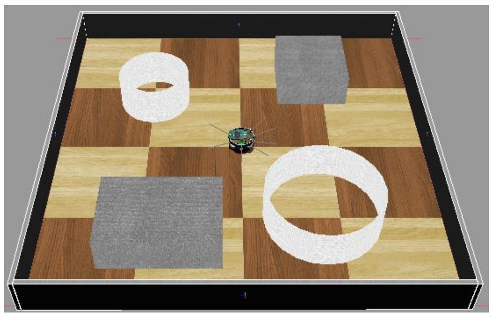


Figure 5 A rectangular environment with four objects is used for this research. The mobile robot at the center is e-puck robot with 8 proximity sensors for robot's observation. The particle filter SLAM algorithm is implemented as the robot's controller

$$RMSE(x_k) = \left[\frac{1}{N}\sum_{i=1}^{N}\left(x_k - \hat{x}_k^i\right)^2\right]^{\frac{1}{2}} \qquad (15)$$

Figure 6 shows the result of the maps obtained from the best particle, which is particle with the highest weight. The leftmost column is map obtained using perfect localization (i.e. the ground truth data from the robot simulator), and the subsequent columns are map obtained from the best particle of the 100 particles, 300 particles and 500 particles respectively. The green dots represent the particles' distribution. We can observed that the map obtained by using 500 particles are better than the map obtained by using 100 particles and 300 particles.

To reflect the performance of each set of particles, the RMSE values at each time step is observed and plotted in RMSE graph shown in Figure 7. The RMSE of 100, 300 and 500 particles are plotted in red, green and blue line respectively. It can be seen that the RMSE keep on increasing due to accumulated error from robot's odometry. However, at x state (i.e. top row) the set of 500 particles manage to maintain the RMSE at 0.3 compared to the set of 100 particles. Although the RMSE graph indicates a small difference, it shows that the set of 500 particles can approximate state distribution better compared to the set of 100 particles. The y state however shows an opposite result. This might be due to translation error, α_1, and rotation error, α_2, need to be adjusted accordingly.

Although by using 500 particles, we can approximate state distribution better, the drawback is the computation taken was much longer since we have to compute the prediction and correction step for each particle.

6.0 CONCLUSION AND FUTURE WORKS

From the results shown in Section 4, the number of particles does affect the resulting map overtime. In this work, the increasing uncertainty caused by accumulation error from noise in wheel encoder and sensor measurements in the robotic system is modeled using set of particles. The drawback of particle filter algorithm is computation cost of the algorithm increases with the number of particles used.

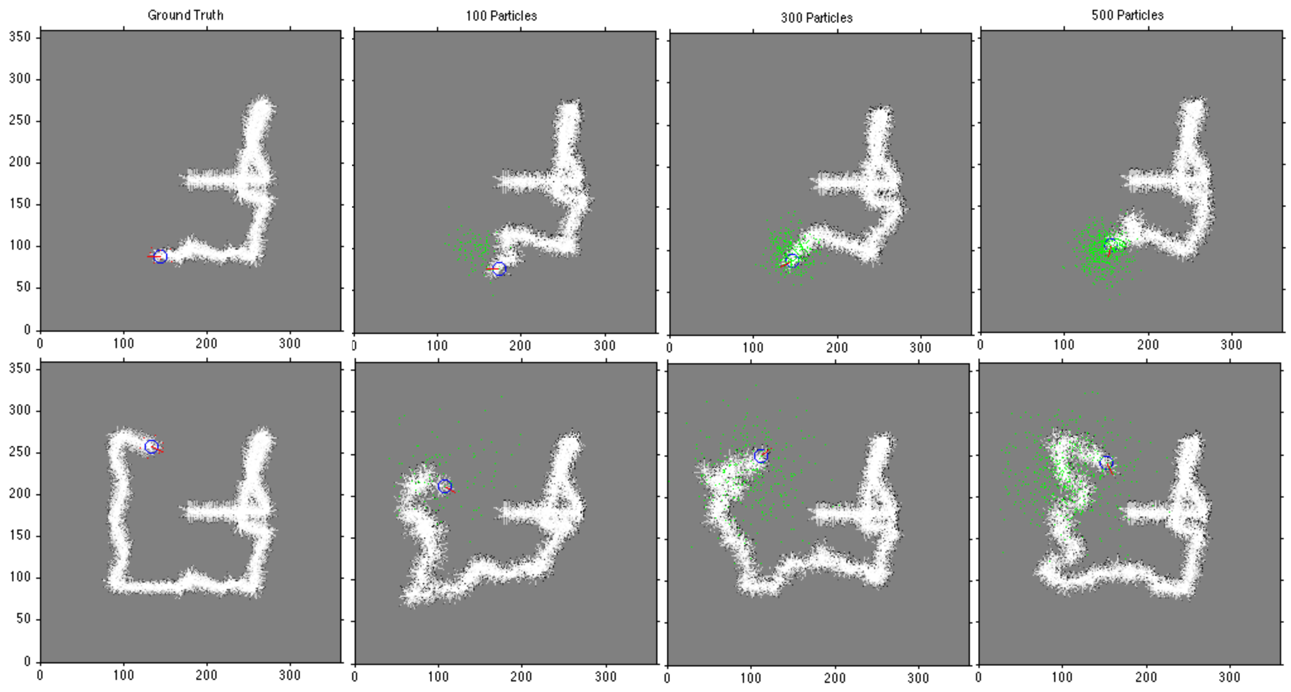


Figure 6 The maps of the rectangular environment obtained at time step 400 (top row) and time step 600 (bottom row). The leftmost is the ground truth data from the robot simulator, and the subsequent columns are map obtained from the best particle of the 100 particles, 300 particles and 500 particles respectively. The green dots represent the particles' distribution

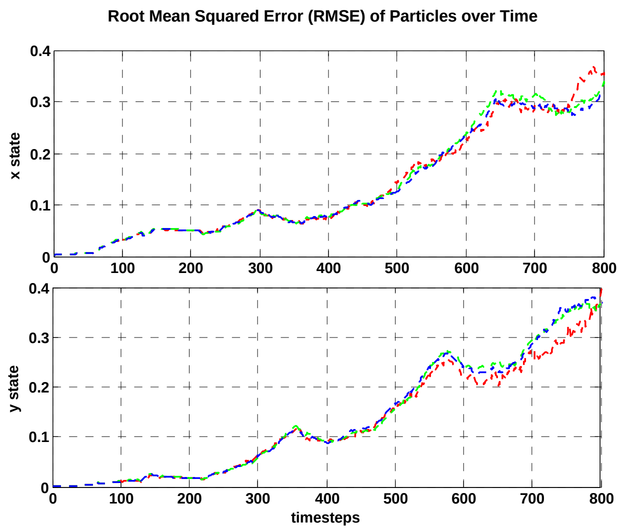


Figure 7 The RMSE graph obtained, zooming in at time step, $t < 800$. The red, green and blue lines represent the RMSE of 100 particles, 300 particles and 500 particles respectively. The top graph is for robot state, x position and the bottom graph for y position

To learn a new map efficiently, the major challenge is to reduce the number of particle used. Another issue is particle depletion problem where the number of unique particles reduces overtime. Thus, the diversity of particles decreases. In this work, the effect of correction step is not clearly visible yet. However, the implementation of particle filter SLAM using a simple differential drive mobile robot, e-puck is accomplished, despite of the difficult SLAM problem dealt with. Improvement will be done to the particle filter SLAM algorithm in order to obtain a more accurate occupancy grid map by utilizing a simple differential drive mobile robot with low-resolution sensor such as the e-puck mobile robot.

Acknowledgement

The authors would like to acknowledge and express much appreciation to the Universiti Teknologi MARA (UiTM), Shah Alam, Universiti Teknikal Malaysia Melaka (UTeM) and Ministry of Education Malaysia (MoE) for the opportunity, facilities and funds to carry out this research.

Reference

[1] K. R. Beevers and W. H. Huang. 2006. SLAM with Sparse Sensing. In *Robotics and Automation, ICRA 2006. Proceedings 2006 IEEE International Conference on, 2006*. 2285-2290.

[2] T. N. Yap and C. R. Shelton. 2009. SLAM in Large Indoor Environments with Low-Cost, Noisy, and Sparse Sonars. In *IEEE International Conference on Robotics and Automation* (ICRA). 1395-1401.

[3] T. Reineking and J. Clemens. 2013. Evidential FastSLAM for grid mapping. In Information Fusion (FUSION). *2013 16th International Conference on*. 789-796.

[4] J.-S. Lee, C. Kim, and W. K. Chung. 2010. Robust RBPF-SLAM Using Sonar Sensors in Non-Static Environments. In *IEEE International Conference on Robotics and Automation (ICRA), 2010*. 250-256.

[5] S. Jo, H. Choi, and E. Kim. 2012. Ceiling Vision Based SLAM Approach Using Sensor Fusion of Sonar Sensor and Monocular Camera. In *Control, Automation and Systems (ICCAS), 2012 12th International Conference on.* 1461-1464.

[6] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe. 2011. A Review of Recent Developments in Simultaneous Localization and Mapping. *2011 6th Int. Conf. Ind. Inf. Syst.* 477-482.

[7] A. Doucet, N. De Freitas, K. Murphy, and S. Russell. 2000. Rao-Blackwellised Particle Filtering for Dynamic Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. 176-183.

[8] K. Murphy. 2000. Bayesian Map Learning in Dynamic Environments. *Adv. Neural Inf. Process. Syst*. 12: 1015-1021.

[9] M. Montemerlo. 2003. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association. Carnegie Mellon University, Pittsburgh.

[10] D. Roller, M. Montemerlo, S. Thrun, and B. Wegbreit. 2003. Fastslam 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping That Provably Converges. In *Proceedings of the International Joint Conference on Artificial Intelligence*,

[11] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. 2003. An Efficient Fastslam Algorithm for Generating Maps of Large-Scale Cyclic Environments from Raw Laser Range Measurements. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003)*. 1: 206-211.

[12] A. Eliazar and R. Parr. DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks. In *IJCAI.* 3: 1135-1142.

[13] G. Grisetti. 2005. Improving Grid-based Slam with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *International Conference on Robotics and Automation*. April: 32-37.

[14] C. Stachniss, G. Grisetti, and W. Burgard. 2005. Recovering Particle Diversity in a Rao-Blackwellized Particle Filter for SLAM After Actively Closing Loops. In *International Conference on Robotics and Automation*. April: 55-60.

[15] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi. 2007. Fast and Accurate SLAM with Rao–Blackwellized Particle Filters. *Rob. Auton. Syst.* 55(1): 30-38.

[16] P. S. Maybeck. 1982. *Stochastic Models, Estimation, and Control*. 3. Academic Press.

[17] R. Siegwart and I. R. Nourbakhsh. 2004. *Introduction to Autonomous Mobile Robots*. 23.

[18] H. P. Moravec. 1988. Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*. 9: 61-74.

[19] S. Thrun. 2003. Learning Occupancy Grid Maps with Forward Sensor Models. *Auton. Robots.* 15(2): 111-127.

[20] C. Schröter, H.-J. Böhme, and H.-M. Gross. 2007. Memory-Efficient Gridmaps in Rao-Blackwellized Particle Filters for SLAM using Sonar Range Sensors. In *EMCR.*

[21] S. Thrun, W. Burgard, and D. Fox. 2005. *Probabilistic Robotics.* MIT Press.