Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

PSO-FastSLAM: An Improved FastSLAM Framework using Particle Swarm Optimization

Heon-Cheol Lee, Shin-Kyu Park, Jeong-Sik Choi, *Student Member*, *IEEE*, and Beom-Hee Lee, *Fellow*, *IEEE*
School of Electrical Engineering and Computer Sciences
Seoul National University
Seoul, Korea
{restore98, mugcup0828, jsforce2, bhlee}@snu.ac.kr

***Abstract*— FastSLAM is a framework which solves the problem of simultaneous localization and mapping using a Rao-Blackwellized particle filter. Conventional FastSLAM is known to degenerate over time in terms of accuracy due to the particle depletion in resampling phase. In this work, a new FastSLAM framework is proposed to prevent the degeneracy by particle cooperation. First, after resampling phase, a target that represents an estimated robot position is computed using the positions of particles. Then, particle swarm optimization is performed to update the robot position by means of particle cooperation. Computer simulations revealed that the proposed framework shows lower RMS error in both robot and feature positions than conventional FastSLAM.**

***Keywords*— mobile robot, FastSLAM, particle filter, particle swarm optimization**

I. Introduction

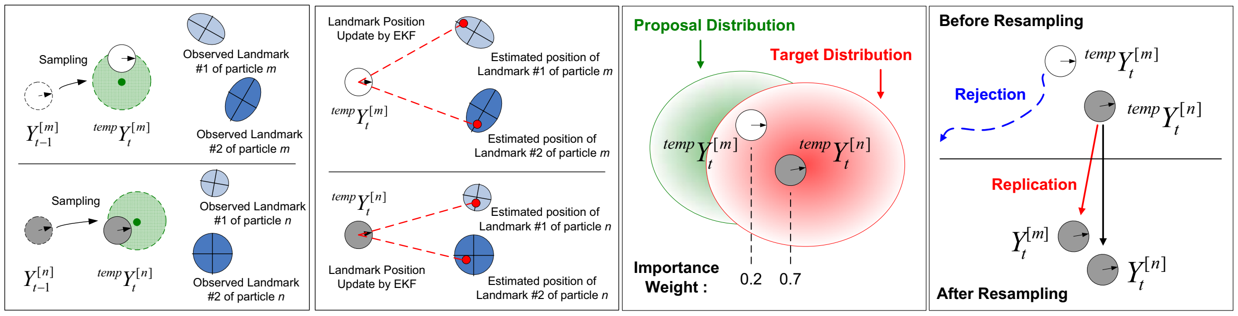
The simultaneous localization and mapping (SLAM) is a fundamental problem of robots to perform autonomous tasks, such as navigation in unknown environments. FastSLAM is a successful algorithm for solving the SLAM problem. Rao-Blackwellized particle filter (RBPF) used in FastSLAM has two advantages over any other algorithms in solving the SLAM problem: linear time complexity and multi-hypothesis data association [1]. However, in the literatures [2], FastSLAM has been noted to degenerate over time in terms of accuracy. This degeneracy happens when a particle set estimating the pose of the robot loses its diversity. There are two main reasons for losing particle diversity in FastSLAM. First, sample impoverishment, caused by mismatch between target distribution and proposal distribution, produces improbable particles which estimate robot pose inaccurately. Second, during the resampling process in FastSLAM, the improbable particles are thrown away and only particles with high weights survive. However, if particles which estimate robot pose accurately get low weight and dumped out, the robot information stored within the dumped particle cannot be recovered. In other words, the miscomputation of particle's weight causes particle depletion problem.

In recent years, many algorithms [2-6] have been proposed to overcome the particle depletion problem. Kwak et al. [3] analyzed several particle filters used in FastSLAM and proposed a compensation technique for dealing with the depletion problem. Kim et al. [4] proposed unscented FastSLAM which uses unscented transform for particle filter, feature initialization, and feature estimator. Similarly, Cugliari et al. [5] used the unscented transform to improve the accuracy of the particle filter and feature estimator, and they also proposed an adaptive selective resampling technique. In our previous work, a new resampling technique is proposed using geometric relation between particles to restrain the particle depletion [6]. So far, these techniques have shown better performance than conventional FastSLAM. However, these methods are not a cooperative way of handling particles in the depletion problem that they merely try to maintain diversity to improve FastSLAM algorithm.

In this work, the concept of particle cooperation in swarm intelligence is introduced to improve the performance of FastSLAM. Because particles cooperate with one another to track a target such as robot position, they can estimate the target more accurately. Therefore, the number of highly probable particles that estimate actual robot pose becomes larger, and the particle depletion problem can be dealt by preventing the improper rejection of particles. There are a few attempts that use swarm intelligence to solve the localization and SLAM problems. Kwok et al. [7] used particle swarm optimization (PSO) to estimate a robot's location. Tdor et al. [8] first introduced the PSO to the SLAM problem. It was a challenging attempt to apply swarm intelligence to the SLAM problem, but it allowed too many random variables and was not performed in a clear framework.

We propose an improved FastSLAM framework, named PSO-FastSLAM, using PSO as the particle cooperation technique for estimating the robot position with FastSLAM. First, as a target for PSO, the most probable location of a robot is obtained using the positions of particles. Then as the particle cooperation method, PSO is used for particles to be gathered around the target. As a result, the number of particles estimating the robot position accurately becomes larger, which means the degeneracy of FastSLAM can be alleviated.

This paper is organized as follows: Section II describes the framework of conventional FastSLAM and the particle depletion problem. In Section III, PSO-FastSLAM is stated with a brief explanation of PSO. Section IV shows the evaluation of the proposed framework using computer simulations. Finally, Section V gives conclusions.



Sampling
$Y_{t-1}^{[m]}$
$^{temp}Y_t^{[m]}$
Observed Landmark #1 of particle *m*
Observed Landmark #2 of particle *m*
Sampling
$Y_{t-1}^{[n]}$
$^{temp}Y_t^{[n]}$
Observed Landmark #1 of particle *n*
Observed Landmark #2 of particle *n*
Landmark Position Update by EKF
Estimated position of Landmark #1 of particle *m*
Estimated position of Landmark #2 of particle *m*
Estimated position of Landmark #1 of particle *n*
Estimated position of Landmark #2 of particle *n*
Landmark Position Update by EKF
Proposal Distribution
Target Distribution
Importance Weight :
0.2
0.7
Before Resampling
Rejection
Replication
$Y_t^{[m]}$
$Y_t^{[n]}$
After Resampling

(a) Sampling (b) Measurement update (c) Importance weight (d) Resampling

Figure 1. A graphical procedure of conventional FastSLAM algorithm (FastSLAM 2.0) (a) Pose sampling of two particles using control input and current measurement where the superscript tmp means a particle is included in a temporary particle set, (b) Measurement update, which is performed per particle, (c) Importance weight, and (d) Resampling; particles are replicated or rejected.

II. Problem Formulation

A. Conventional FastSLAM Framework

The key mathematical insight of conventional FastSLAM is the fact that the full SLAM posterior can be factored as [1]:

$$p(x_{1:t}, M \mid z_{1:t}, u_{1:t}, c_{1:t}) \\ = p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t}) \prod_{n=1}^{N_f} p(m_n \mid x_{1:t}, z_{1:t}, u_{1:t}, c_{1:t}) \tag{1}$$

where $c_{1:t}$=$c_1,\ldots,c_t$ are the correspondences, and $x_{1:t}$ is the robot path from the start up to time t, and M is a global map, and m_n is a local map of the n-th particle. $z_{1:t}$ and $u_{1:t}$ are the measurements and controls up to time t, respectively. FastSLAM uses particle filter to compute the posterior over robot paths, denoted by $p(x_{1:t} \mid z_{1:t}, u_{1:t}, c_{1:t})$. For each feature in the map, FastSLAM uses a separate estimator over its location $p(m_n \mid x_{1:t}, z_{1:t}, c_{1:t})$, one for each n = 1, ..., N_f where N_f is the number of features. The feature estimators are conditioned on the robot path, which means there are separate copies of each feature estimator, one for each particle. More precisely, feature locations are estimated using EKFs. Due to the factorization, FastSLAM maintains a separate EKF for each feature, which makes the update more efficient than that in EKF-SLAM in terms of computational complexity. By keeping the feature estimates independently, FastSLAM avoids the quadratic cost of computing a joint map covariance matrix.

A particle at time t, $Y_t^{[k]}$ in FastSLAM is denoted by

$$Y_t^{[k]} = \left\langle x_t^{[k]}, \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]}, \ldots, \mu_{N_f,t}^{[k]}, \Sigma_{N_f,t}^{[k]} \right\rangle \tag{2}$$

where the [k] indicates the index of the particle, and $x_t^{[k]}$ is the pose estimate of the robot at time t. Only the most recent pose $x_t^{[k]}$ is used in FastSLAM, so a particle keeps only the most recent pose. $\mu_{n,t}^{[k]}$, $\Sigma_{n,t}^{[k]}$ are mean and covariance of the Gaussian, representing the n-th feature location relative to the k-th particle, respectively. Altogether, these elements form the k-th particle, $Y_t^{[k]}$, and there are total N_p particles and N_f feature estimates in a particle set.

A simple graphical procedure of conventional FastSLAM (FastSLAM 2.0) algorithm is depicted in Fig. 1. For convenience, only two particles among a particle set are shown. In Fig. 1(a), each particle samples a robot pose based on the proposal distribution which takes the measurement z_t into account. All of the sampled poses constitute a temporary particle set. Then, each particle updates the posterior over the feature estimates based on the measurement z_t and the sampled pose $^{tmp}x_t^{[k]}$ as shown in Fig. 1(b). The next step is to compute the importance weight of k-th particle using the following weighting function quotient as shown in Fig. 1(c):

$$w_t^{[k]} = \frac{\text{target distribution}}{\text{proposal distribution}}. \tag{3}$$

The last process is resampling, which draws N_p particles from the temporary particle set. As shown in Fig. 1(d), the temporary particle with high importance weight, $Y_t^{[n]}$ is replicated three times whereas the one with low importance weight, $Y_t^{[m]}$ is rejected or thrown away in the resampling process. This means that the information on robot path and feature estimates of the rejected particles is lost.

B. Particle Depletion Problem

The rejection of bad particles that estimate robot pose inaccurately is indispensable in resampling phase. However, the rejection of the bad particles requires carefulness of conduct because it may deteriorate the multi-hypothesis data association that is the strong point of FastSLAM over other SLAM solutions. As time passes, only particles with high weights survive and perform replication, and particles with low weights disappear together with their information on robot pose and the locations of features. Therefore, the number of distinct estimates of a robot and features decreases, which means the diversity of particles decreases.

This problem so called particle depletion problem is inevitable unless the sufficient number of good particles that estimate accurate robot pose is maintained. However, the maintenance is not easy because of the lack of knowledge about the location of a robot, and only when particles share and use the knowledge of the most probable location of a robot, the sufficient number of good particles can be maintained.

III. PSO-FastSLAM

A. Particle Swarm Optimization (PSO)

Recently, the idea that particles cooperate with one another to track a common target has been applied to various fields such as mathematics, sciences, and engineering. PSO is a population-based technique, similar in some respects to evolutionary algorithms, except that potential solutions (particles) move and cooperate with one another, rather than evolve through the search space. The particle dynamics which govern the movement are inspired by models of swarming and flocking [10]. The core principals of PSO with respect to particle cooperation are described in Fig. 2.

In PSO, each particle has a position, a velocity, and memory toward two attractors. One is the best position attained by that particle so far (personal or particle attractor). The other is the best of the particle attractors in a certain neighborhood (global or swarm attractor). Therefore, the swarm attractor enables particles to share information, while the particle attractors serve as individual particle memories. A particle *i* is defined by its position $\vec{x}_i$, the position of its personal best solution found so far, $\vec{p}_i$, and its velocity $\vec{v}_i$. Furthermore, each particle knows the best solution found so far by any of its neighbors. Different particle topologies are explored in [12], but the standard neighborhood is global (*gbest*), i.e., all particles know the position of the globally best solution found so far $\vec{p}_g$. In the literature [11], particle positions and velocities are generated randomly. The algorithm proceeds iteratively, updating first all velocities, and then all particle positions as follows:

$$\vec{v}_i(s+1) = \chi[\vec{v}_i(s) + c_1\vec{\varepsilon}_1 \cdot (\vec{p}_g - \vec{x}_i(s)) + c_2\vec{\varepsilon}_2 \cdot (\vec{p}_i - \vec{x}_i(s))] \quad (4)$$

$$\vec{x}_i(s+1) = \vec{x}_i(s) + \vec{v}_i(s+1) \quad (5)$$

where s=1,2,…,N_e and N_e is the number of epochs. The constriction factor $\chi < 1$ acts like friction, slowing the movement of particles, so that finer exploration is achieved. c_1 and c_2 control the relative attraction to the global best and personal best found solutions, respectively. Finally, $\vec{\varepsilon}_1$ and $\vec{\varepsilon}_2$ are vectors of random variables drawn with uniform probability from 0 to 1.

B. PSO based Particle Filter (PSO-PF)

Recently, Zhang et al. [9] proposed PSO-PF which combines generic particle filtering algorithm with standard PSO. In their work, after resampling in the generic particle filter, PSO is preformed using the weight of a particle as a fitness value. The most weighted particle, therefore, is used as a global attractor (target) in every iteration step. Applying PSO in the particle filter allows the particles to maintain diversity and makes them to move toward the most weighted particle.

In our work, similar to PSO-PF, we used the idea of PSO in FastSLAM algorithm; however, instead of selecting the most weighted particle as the target, we select the mean of the particles' positions for the target's position. According to the recent works in particle filter-based SLAM, the mean of the particles' positions is much closer to the actual robot position than the position of the most weighted particle [17, 18]. This is the most important aspect when PSO is applied in SLAM because if a poorly estimated particle is determined as the target then even though particles maintain diversity, they will be optimized around the incorrect target. Therefore more accurate SLAM results, i.e. robot's location and map, will be obtained by using the mean of the particles' positions as the target in PSO. For each particle *i*, the personal best of PSO is calculated by

$$\vec{p}_i = \arg\min_{\vec{x}_i(s)} \|\vec{x}_i(s) - m_{\vec{x}}\|, \ \textit{where } m_{\vec{x}} = \frac{1}{N_p} \cdot \sum_{i=1}^{N_p} \vec{x}_i(1) \quad (6)$$

where s=1,2,…,N_e, and $\vec{x}_i(1)$ represents the original position of the *i*-th particle at time step *t*, $x_t^{[i]}$. And the global best of PSO at the *s*-th epoch is calculated by

$$\vec{p}_g = \arg\min_{\vec{x}_i(s)} \|\vec{x}_i(s) - m_{\vec{x}}\| \quad (7)$$

where i=1,2,…,N_p. The use of PSO in FastSLAM has, other than maintaining diversity, two advantages over generic FastSLAM algorithm. First, it is possible to be recovered from miscomputation of new particles in the sampling phase. A particle's personal best in PSO is chosen to be the particle's position at the moment when the particle has highest weight; therefore, the particle will be recovered by (4) and (5). Also, by selecting the position of the closest particle to the mean of particles' positions as the global best, the particles can be located near the actual robot position. The whole algorithm of the modified PSO-PF is shown in Fig. 3. Unlike in PSO-PF, the PSO-FastSLAM performs an additional step of computing the target.

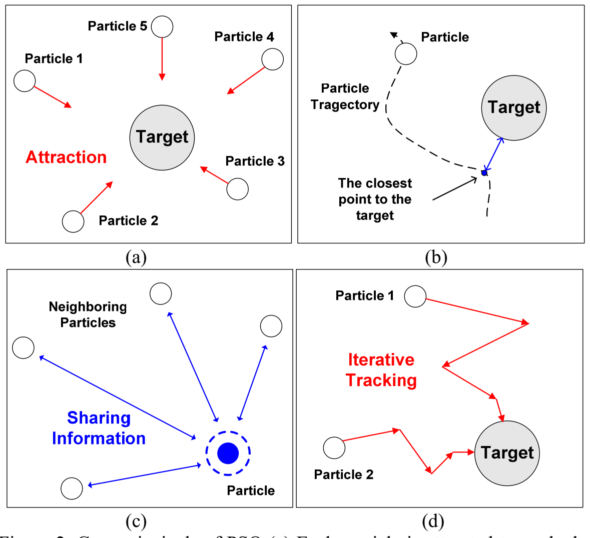


Figure 2. Core principals of PSO (a) Each particle is attracted towards the location of the target. (b) Each particle 'remembers' where it was closer to the target. (c) Each particle shares information with its neighbors (originally, all other particles) about its closest location to the target. (d) Each particle tracks the target iteratively using its own and shared information.

Algorithm 1 Modified PSO-PF

```
1:  // Conventional Particle Filter
2:  for $m = 1$ to $N_p$
3:      retrieve m-th particle from particle set
4:      do Sampling [described in Fig. 1(a)]
5:      do Measurement Update [described in Fig. 1(b)]
6:      do Importance Weight [described in Fig. 1(c)]
7:      do Reampling [described in Fig. 1(d)]
8:  end
9:  // Particle Cooperation
10: Compute the target : Mean of positions of particles
11: for $m = 1$ to $N_p$
12:     Initialize the velocity of the $m$-th particle
13:     Initialize the position of the $m$-th particle
14:     Initialize the fitness value of the $m$-th particle
15: end
16: for $k = 1$ to $N_E$
17:     for $m = 1$ to $N_p$
18:         Compute the fitness value of the $m$-th particle
19:     end
20:     Determine the global best [Eq. (7)]
21:     for $m = 1$ to $N_p$
22:         Determine the personal best of the $m$-th particle
             [Eq. (6)]
23:         Compute the $m$-th particle's velocity [Eq. (4)]
24:         Compute the $m$-th particle's position [Eq. (5)]
25:     end
26: end
```

Figure 3. The modified PSO-PF consists of two parts. The first part is equal to the generic particle filter. The second part is the particle cooperation which uses PSO. Unlike PSO-PF, the modified PSO-PF performs an additional step of computing the target.

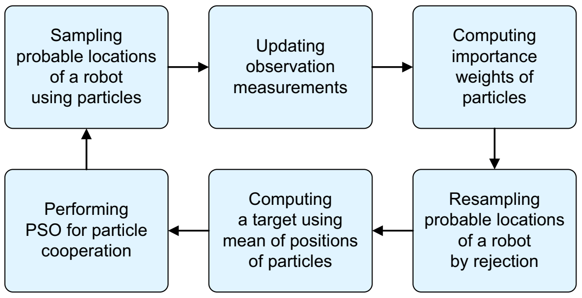

Sampling probable locations of a robot using particles
Updating observation measurements
Computing importance weights of particles
Resampling probable locations of a robot by rejection
Computing a target using mean of positions of particles
Performing PSO for particle cooperation

Figure 4. Block diagram of the PSO-FastSLAM. Unlike conventional FastSLAM, PSO-FastSLAM performs an additional process which consists of target computation and particle cooperation.

C. *PSO-FastSLAM Framework*

PSO-FastSLAM differs from conventional FastSLAM because of an additional process for compensating the particle depletion. The compensation process which is the key strategy of PSO-FastSLAM attracts the particles toward the actual robot position while maintaining particle diversity. The overall block diagram of PSO-FastSLAM is shown in Fig. 4. After resampling probable locations of a robot by rejection, a target is computed using mean of positions of particles. Then PSO is performed for particle cooperation in PSO-FastSLAM.

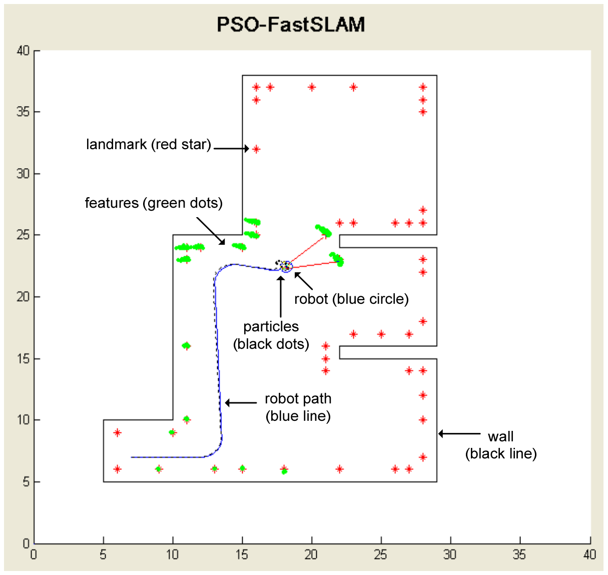


Figure 5. Simulation environment. The robot path is given, but the robot cannot follow the path due to the motion noise. Feature errors were getting larger at point A, and the inner loop-closure occurred at point B and the outer loop-closure at point C.

IV. SIMULATIONS

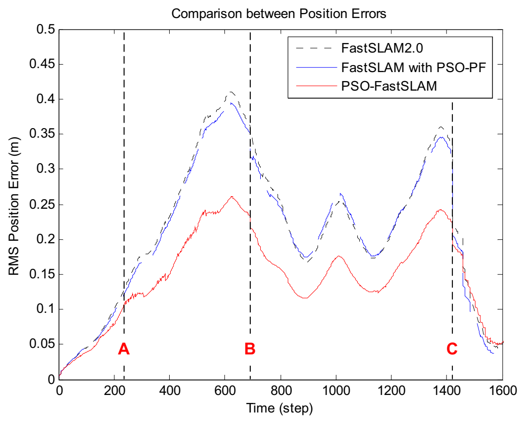
A. *Simulation Setups*

Bailey developed the FastSLAM simulator [16] which is opened to the public. In the literature [2], the consistency of the FastSLAM2.0 was discussed using the simulator. In the same simulation environment, PSO-FastSLAM was implemented and results are compared with other FastSLAM frameworks.

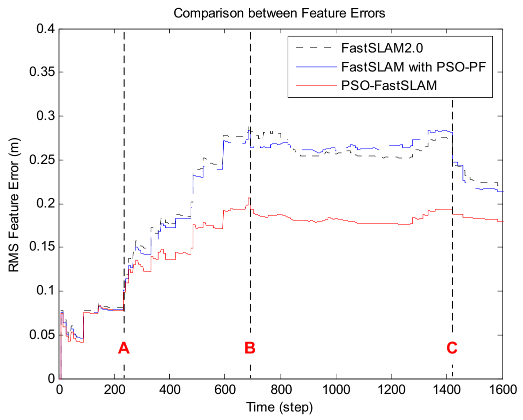
In order to perform realistic simulations, the simulations were performed in an environment with a sparse map with 50 landmarks as shown in Fig. 5. The global map of the environment is 24 meter by 33 meter with several convex and concave boundaries to give robots rotations that may cause the errors in odometry sensor. Because the actual velocities differ from the commanded ones or measured ones, we modeled this difference by a zero-centered random variable with finite variance [15]. More precisely, we assumed the actual velocities are given by

$$\begin{pmatrix} \hat{v} \\ \hat{w} \end{pmatrix} = \begin{pmatrix} v \\ w \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1|v|+\alpha_2|w|} \\ \varepsilon_{\alpha_3|v|+\alpha_4|w|} \end{pmatrix} \tag{8}$$

where v and w are translational and rotational velocities of the robot, respectively, and $\hat{v}$ and $\hat{w}$ are their estimates taking into account error ε_b, a zero-mean error variable with standard deviation b. The parameters from α_1 to α_4 are robot-specific parameters that represent the accuracy of the odometry sensor. The less accurate the odometry sensor is, the larger their parameters are. In this paper, the variance of the normal distribution of the translational and rotational velocities was set to ($0.1\text{m}^2/\text{s}^2$, $1°/\text{s}^2$) by adjusting the parameters.



(a) Position errors per simulation run



(b) Feature errors per simulation run

Figure 6. RMS Estimation errors at every time step. The performance of the FastSLAM with PSO-PF is similar to that of conventional FastSLAM (FastSLAM2.0). Clearly, the performance of the PSO-FastSLAM is better than the two cases.

The landmark measurements are also dependent on the robot velocities and can be modeled by

$$\begin{pmatrix} \hat{r} \\ \hat{\theta} \end{pmatrix} = \begin{pmatrix} r \\ \theta \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_5|v|+\alpha_6|w|} \\ \varepsilon_{\alpha_7|v|+\alpha_8|w|} \end{pmatrix} \qquad (9)$$

where r is the distance between the robot and the observed landmark, and θ is the orientation of the observation measurements in local coordinates of the robot. $\hat{r}$ and $\hat{\theta}$ are their estimates taking into account error ε_b , a zero-mean error variable with standard deviation b. The parameters from α_5 to α_8 are sensor-specific parameters. For instance, a laser range-finger has small errors both in distance and orientation whereas a sonar sensor has a big error in orientation and a small error in distance. In this paper, the variance of the normal distribution of distance and orientation of the landmark measurement was set to (0.1m^2, 1°^{2}) by adjusting the parameters.

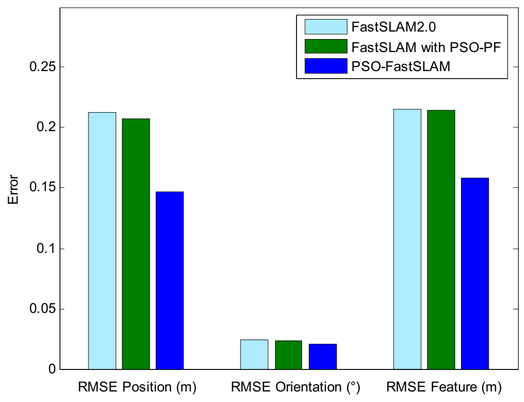


Figure 7. RMSE comparison. For the position and orientation of a robot, the RMSE of PSO-FastSLAM is smaller than other cases. Also, for the features of the built map, the RMSE of PSO-FastSLAM is smaller than other cases.

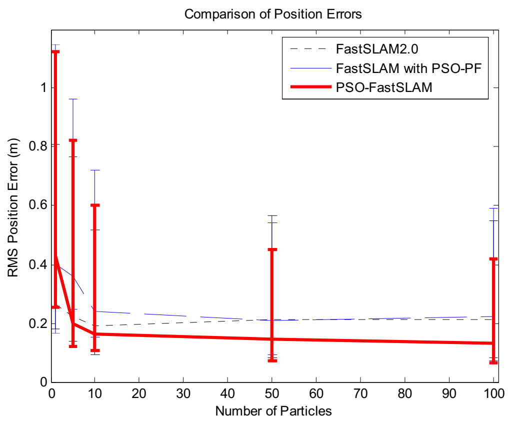
B. *Simulation Results*

To evaluate the performance of PSO-FastSLAM, the root mean square (RMS) errors of position and feature estimation were computed at every time step as shown in Fig. 6. The figure shows the average of the results of 50 simulations in same circumstances. In the simulations, the weights of all particles were initialized with 0.02 after resampling when 50 particles were used. The figure shows that the errors increased rapidly at point A where a robot moves along abrupt curves, and they decreased at point B and C where a robot closed the inner and outer loop of the map. Among three cases, PSO-FastSLAM showed the smallest errors in both of position and feature estimation. For the sake of more clear analysis, the RMS errors in the robot's position and orientation, and features of the map are summarized in Fig. 7. The errors in orientation are similar, but the errors in position and feature estimates of PSO-FastSLAM are much smaller than other FastSLAM algorithms.

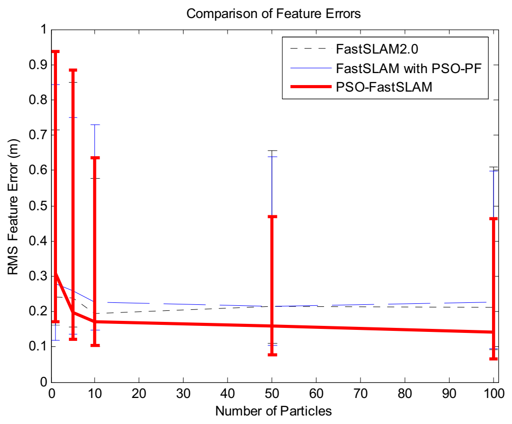
For given a wide range of particle numbers from 1 to 100, the RMS errors in robot position and features were computed. Each simulation was run 50 times, and the averaged results were compared as shown in Fig. 8. The figure shows not only the averaged values but also minimum and maximum values. When more than ten particles are used, the performance of the PSO-FastSLAM is better than that of others apparently. The standard deviations of estimation errors are summarized in Table. I. Each value in the table averages the whole results with different numbers of particles which are shown in Fig. 8. The standard deviations of PSO-FastSLAM were smaller than those of other FastSLAM frameworks.

V. CONCLUSIONS

FastSLAM has been shown to degenerate over time in terms of accuracy due to the particle depletion in resampling process. The problem occurs when particles cannot sufficiently utilize information on robot pose because of the limitation on



(a) Position errors with different numbers of particles



(b) Feature errors with different numbers of particles

Figure 8. RMS Estimation errors with different numbers of particles. Below ten particles, the comparison of position and feature errors is not clear. However, above ten particles, the performance of the PSO-FastSLAM is better than that of others.

TABLE I
STANDARD DEVIATION OF ESTIMATION ERRORS

Type	Position (m)	Feature (m)
FastSLAM 2.0	0.0756	0.0871
FastSLAM with PSO-PF	0.0768	0.0768
PSO-FastSLAM	0.0741	0.0752

data exchange. In this work, PSO-FastSLAM which adopts the PSO to the particle depletion problem was proposed to improve the performance of FastSLAM. Particles are cooperated through PSO after resampling phase in FastSLAM for better estimation. The performance of PSO-FastSLAM was verified by reduced RMS errors in robot pose and features of map in computer simulation. As future work, the robustness of PSO-FastSLAM will be analyzed and verified in various environments.

ACKNOWLEDGMENT

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) NRL Program grant funded by the Korea government (MEST) (No.R0A-2008-000-20004-0) and the Growth Engine Technology Development Program funded by Ministry of Knowledge Economy, the ASRI, and the BK21 Information Technology at Seoul National University.

REFERENCES

[1] M. Montemerlo, “FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association,” Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2003.
[2] T. Bailey, J. Nieto, and E. Nebot, ”Consistency of the FastSLAM algorithm,” in *Proc. IEEE Int. Conf. Robotics and Automation*, 2006, pp.424-429.
[3] N. Kwak, G. W. Kim, and B. H. Lee, "A new compensation technique based on analysis of resampling process in FastSLAM,” *Robotica*, Mar. 2008.
[4] C. Kim, R. Sakthivel, and W. K. Chung, “Unscented FastSLAM: A robust and efficient solution to the SLAM problem,” *IEEE Trans. Robotics*, vol. 24, Aug. 2008.
[5] M. Cugliari and F. Martinelli, “A FastSLAM algorithm based on the unscented filtering with adaptive selective resampling,” *Field and Service Robotics*, vol. 42, pp. 359–368, Jun. 2008.
[6] I. K. Kim, N. Kwak, H. C. Lee, and B. H. Lee, "Improved particle fusing geometric relation between particles in FastSLAM", *Robotica*, doi: 10.1017/S0263574708005250 (Online), Jan. 2009.
[7] N. M. Kwok, D. K. Liu, and G. Dissanayake, “Evolutionary computing based mobile robot localization,” *Engineering Applications of Artificial Intelligence*, vol. 19, pp. 857-868, Dec. 2006.
[8] B. Tdor and D. Darabos, “Simultaneous localization and mapping with particle swarm localization,” in *Proc. IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Sofia, Bulgaria, 2005, pp. 216-221.
[9] G. Zhang, Y. Cheng, F. Yang, and Q. Pan, “Particle filter based on PSO,” in *Proc. Int. Conf. Intelligent Computation Technology and Automation*, 2008, pp. 121-124.
[10] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[11] T. Blackwell and J. Branke, “Multiswarms, exclusion, and anti-convergence in dynamic environments,” *IEEE Trans. Evolutionary Computation*, vol. 10, pp. 459-472, Aug. 2006.
[12] S. Janson and M. Middendorf, “A hierarchical particle swarm optimizer for dynamic optimization problems,” in *Applications of Evolutionary Computing*. ser. Lecture Notes in Computer Science, vol. 3005, pp. 513–524, 2004.
[13] N. Kwak, I. K. Kim, H. C. Lee, and B. H. Lee, “Adaptive prior boosting technique for the efficient sample size in FastSLAM,” in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, San Diego, CA, USA, 2007.
[14] A. Doucet, N. Freitas, K. Murphy, and S. Russell, “Rao-Blackwellised particle filtering for dynamic Bayesian networks,” in *Proc. 16th Conf. Uncertainty in Artificial Intelligence*, 2000, pp. 176-183.
[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, Cambridge, U.S.A., 2005, pp. 117-187.
[16] T. Bailey. (2009. Mar. 12). [Online]. Available: http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm
[17] N. Fairfield, G. A. Kantor, and D. Wettergreen, “Three Dimensional Evidence Grids for SLAM in Complex Underwater Environments,” *in proc. 14th International Symposium of Unmanned Untethered Submersible Technology*, 2005.
[18] Nosan Kwak, Beom-Hee Lee, and Kazuhito Yokoi, "Result Representation of Rao-Blackwellized Particle Filtering for SLAM," *International Conference on Control, Automation and Systems 2008*, Oct. 14-17, 2008, Seoul, Korea, pp. 698-703.