The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Multiswarm Particle Filter for Vision Based SLAM

Hee Seok Lee and Kyoung Mu Lee

Abstract— **Particle Filters have been widely used as a powerful optimization tool for nonlinear, non-Gaussian dynamic models such as Simultaneous Localization and Mapping (SLAM) and visual tracking. Particle filters, however, often suffer from particle impoverishment, which is caused by a mismatch between proposal distribution and target distribution. To solve this problem, we propose a new method to improve the efficiency of particle filters by employing the Particle Swarm Optimization (PSO), which is a kind of swarm intelligence algorithm. The PSO, especially its variant for dynamic models, is combined with the generic particle filter to get samples that are well matched with target distribution. The resulting filter is applied to a vision based SLAM system and its performance is tested. We present experimental results that demonstrate improved accuracy in localization and mapping at the same or less computational cost than the conventional particle filters.**

I. Introduction

Particle filters are the most popular optimization tools for current SLAM implementations. To deal with the high dimensionality of SLAM, Murphy [1] and Doucet [2] introduced the Rao-Blackwellized particle filter (RBPF) that reduces the variance of samples by dividing the solution space into two parts, localization part and mapping part. In RBPF, although the required number of particles is reduced by sampling the poses only, the efficiency of the particle filters w.r.t. the number of particles is still the main issue of the particle filter based SLAM approaches.

A good proposal distribution is the key factor to a particle filter's efficiency, and there have been many researches on how to design better proposal distribution of particle filters. Fox [3] and Soto [4] suggested the adaptive methods for economizing particles according to current estimation quality. In [3], the approximation error is measured by the Kullback-Leibler distance, and the number of particles is determined according to the state uncertainty. Soto [4] proposed a method of not only changing the number of particles but also adapting the proposal distribution.

In some cases, observations of range sensors provide more accurate information for the robot pose than motion sensors, so using them can improve the proposal distribution. In a work of Montemerlo *et al.*[5], measurements from range sensors are incorporated with the motion prior for the proposal distribution. They employ the linearization of optimal importance distribution [6] of the robot pose using recent sensor measurements. Usually measurements from the range sensor provide more concentrated distribution for the robot pose, so the proposal distribution using these sensor measurements can reduce the required range of sampling and consequently reduce the number of particles required. However, the Gaussian approximation of robot pose is not ideal, and the linearization of the optimal importance function is not feasible in some cases such as grid mapping [7].

In this paper we present a novel Particle Swarm Optimization (POS)-based particle filter that is accurate and efficient in terms of both speed and the number of particles required. The PSO is an optimization algorithm that is inspired by the swarm intelligence. Similar to the population-based Monte Carlo method optimization approaches, PSO estimates the optimal value of a function using a set of samples. It has a simple mechanism which uses a relationship between particles, and it shows good optimization performances. With a consideration on the dynamic properties of the SLAM problem, it is successfully applied to the particle filter based SLAM and improves the filter's efficiency.

II. Related work

During the last several decades, various algorithms inspired by the swarm intelligence have been developed for the optimization of linear or nonlinear functions in evolutionary computation community [8]. These include Particle Swarm Optimization (PSO) [9], Ant Colony Optimization (ACO) [10], and Bee Algorithm [11]. These algorithms are similar to the particle filters which use a number of samples to estimate the optimal state, but they are different from particle filters in that the swarm intelligence algorithms use relationship between particles while the particle filters do not. PSO, the algorithm we use, models the movement of a flock of birds.

At the start of the algorithm particles are distributed randomly in the problem space, and then slowly they move toward the current local optimum points. This algorithm was introduced for the purpose of optimizing static target functions, but afterwards some methods were introduced to apply the algorithm to dynamic models [12], [13].

There have been researches trying to apply PSO to dynamic problems such as robot localization and visual tracking, where the particle filters have already been applied. Especially, the visual tracking has been a main target of these approaches. Zhang *et al.*[14] applied PSO for each frame and automatically tracked a target object. Tong *et al.*[15] and Wang *et al.*[16] combined PSO with the particle filter and applied it to the visual tracking and global localization of robot. In their algorithms, particles are propagated first by motion prior and then PSO is applied to re-distribute particles to high probability regions. These studies, however,

H. Lee is with the Department of Electrical Engineering and Computer Science, Seoul National University, 599 Gwanangno, Gwanak-gu, Seoul, Korea. ultra21@snu.ac.kr

K. Lee is with Faculty of the Department of Electrical Engineering and Computer Science, Seoul National University, 599 Gwanangno, Gwanak-gu, Seoul, Korea. kyoungmu@snu.ac.kr

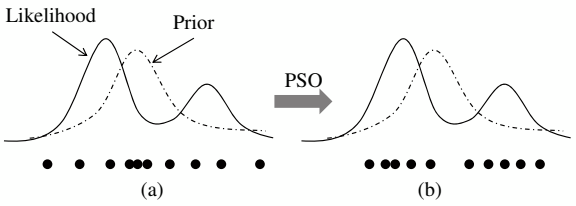


Fig. 1. Distribution before PSO (a) and after PSO (b).

did not present the representation of PSO in the view of particle filters, and also did not consider the characteristics of dynamic environment.

Applying PSO directly to the particle filter can lead to some problems such as lack of particle diversity after PSO procedure has been applied. Once PSO is applied to the particles, then most particles are converged around the local or global maximum point of the likelihood. To solve this problem [14] tried to re-distribute the particles randomly after PSO, but it was not satisfactory.

Some studies have been conducted on PSO for dynamic environment, which we can use to find a solution to the above problems. Blackwell [12] [13] suggested methods such as multiswarm, exclusion, and anti-convergence algorithms to apply the PSO to dynamic environment problems. To deal with the lack of particle diversity, they introduced the concept of repulsion between particles like repulsion between electrons. Additionally, to maintain multiple local maxima they proposed a mechanism for multiple swarm management. These algorithms prevent the over fitting of particles to the current measurements, and the algorithm can be robust to the instant failure of finding optima.

III. Multiswarm particle filter

We maintain this particle filter-based framework and combine the PSO for better particle generation. Similar to the observation incorporated methods for the improved proposal, our PSO combined method provides an effective way to get samples matched with the observations. Since we deal with a dynamic system, we apply the PSO that is designed for a dynamic environment. It includes two concepts: the use of quantum particles, and the use of multiple swarms. Quantum particles are non-converging particles, which are used to provide the diversity of particles. Multiple swarms enable us to track the multiple hypotheses of the robot trajectories. It is useful in situations where it is difficult to determine the robot pose with current observations. The resulting particle filter generates particles with a high likelihood as the original PSO wants to do, but it does not lose the particle diversity. The algorithm will be described in the next subsection.

A. Particle Swarm Optimization

The process of basic single swarm PSO algorithm is as follows. At time t, particles are initially distributed into a problem space with a prior distribution and the PSO starts. Through the PSO iteration, all particles' positions and velocities are updated to find better optimal points. At kth iteration, the particle i has its position $\mathbf{x}_t^{(i,k)}$ ($\mathbf{x}_t^{(i,k)}$ is the robot pose in SLAM) and its velocity $\mathbf{v}_t^{(i,k)}$, and its best position $\mathbf{x}_{tb}^{(i)}$ from its history. Velocity indicates where to go, which is estimated by the relative positions between the particle's current position and its best record $\mathbf{x}_{tb}^{(i,k)}$, and the swarm's current optimum $\mathbf{g}_t$. At kth PSO iteration, the position and velocity of ith particle is updated by

$$\begin{aligned}\mathbf{x}_t^{(i,k+1)} &= \mathbf{x}_t^{(i,k)} + \mathbf{v}_t^{(i,k)} \\ \mathbf{v}_t^{(i,k+1)} &= w\mathbf{v}_t^{(i,k)} + c_1 r_1(\mathbf{x}_{tb}^{(i)} - \mathbf{x}_t^{(i,k+1)}) + c_2 r_2(\mathbf{g}_t - \mathbf{x}_t^{(i,k+1)}),\end{aligned} \tag{1}$$

where w is a constant corresponds to the inertia, and c_1, c_2 are also constant weights for the relative positions. $r_1, r_2 \in [0,1]$ are random values generated for each particle and each PSO iteration. The current best position $\mathbf{x}_{tb}^{(i)}$ and the swarm's current optimum $\mathbf{g}_t$ are updated by every particle's update. With this procedure, the particles explore the space around the current optimum and try to find better optimal point.

B. Swarm Optimization Combined Particle filter

The posterior distribution of the robot trajectories and landmark positions is formulated by

$$p(\mathbf{r}_{1:t}, \mathbf{m} | \mathbf{z}_{1:t}, \mathbf{u}_{0:t-1}) = p(\mathbf{m} | \mathbf{r}_{1:t}, \mathbf{z}_{1:t}) p(\mathbf{r}_{1:t} | \mathbf{z}_{1:t}, \mathbf{u}_{0:t-1}), \tag{2}$$

where $\mathbf{r}_{1:t}$ and $\mathbf{m}$ are the trajectories of the robot from time 1 to t, and the estimated map, respectively. $\mathbf{u}_{0:t-1}$ and $\mathbf{z}_{1:t}$ are the odometry and the observations respectively. Following the Rao-Blackwellized particle filter (RBPF) based framework [1], we first sample the robot pose from the proposal distribution $\pi(\mathbf{r}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{r}_{1:t-1})$, and then estimate the positions of the landmarks using the sampled robot pose via EKF for each particle. For the particle i which has the robot pose $\mathbf{r}_t^{(i)}$ at time t, the importance weight of the particle is evaluated by

$$w_t^{(i)} = w_{t-1}^{(i)} \frac{p(\mathbf{z}_t | \mathbf{r}_t^{(i)}) p(\mathbf{r}_t^{(i)} | \mathbf{r}_{t-1}^{(i)}, \mathbf{u}_{t-1})}{\pi(\mathbf{r}_t^{(i)} | \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}, \mathbf{r}_{1:t-1}^{(i)})}. \tag{3}$$

If the motion prior is noisy due to the motion sensor's error or due to external forces, then the proposal distribution of the robot pose is not coincident with the likelihood function. Fig. 1 (a) shows this situation, known as the particle degeneracy problem. To deal with this problem, we apply the PSO after particle propagation to adapt the particles to the likelihood distribution. We can divide the sampling procedure into two parts: one is a transition by motion prior $\mathbf{u}_t$ and the other is a transition by PSO dynamics given by equation (1). In the PSO procedure, each particle moves K times, where K is the number of PSO iterations. We represent the intermediate position of the particle i at kth PSO iteration as $\mathbf{r}_t^{(i,k)}$. $\mathbf{r}_t^{(i,K)} = \mathbf{r}_t^{(i)}$ is the final obtained position of the ith particle at time t. $\mathbf{r}_t^{(i,k)}$ corresponds to $\mathbf{x}_t^{(i,k)}$ in equation (1). Then we set the proposal distribution

$\pi(\mathbf{r}_t^{(i)}|\mathbf{z}_{1:t},\mathbf{u}_{1:t-1},\mathbf{r}_{1:t-1}^{(i)})$ as

$$\pi(\mathbf{r}_t^{(i)}|\mathbf{z}_{1:t},\mathbf{u}_{1:t-1},\mathbf{r}_{1:t-1}^{(i)}) = p(\mathbf{r}_t^{(i,0)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})p(\mathbf{r}_t^{(i,1)}|\mathbf{z}_t,\mathbf{r}_t^{(i,0)}) \cdots p(\mathbf{r}_t^{(i)}|\mathbf{z}_t,\mathbf{r}_t^{(i,K-1)}). \tag{4}$$

$p(\mathbf{r}_t^{(i,0)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})$ is the probability of transition by the odometry and $p(\mathbf{r}_t^{(i,k+1)}|\mathbf{z}_t,\mathbf{r}_t^{(i,k)})$ is the probability of particle position update at each kth PSO iteration. According to this proposal, we first propagate the particles from the motion probability $p(\mathbf{r}_t^{(i,0)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})$ and update the initial distribution through the PSO dynamics with the probability $p(\mathbf{r}_t^{(i,k+1)}|\mathbf{z}_t,\mathbf{r}_t^{(i,k)})$. In the PSO dynamics, particles move toward the high likelihood region, so we can approximate the probability $p(\mathbf{r}_t^{(i,k+1)}|\mathbf{z}_t,\mathbf{r}_t^{(i,k)})$ as

$$p(\mathbf{r}_t^{(i,k+1)}|\mathbf{z}_t,\mathbf{r}_t^{(i,k)}) \simeq \frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i,k+1)})}{p(\mathbf{z}_t|\mathbf{r}_t^{(i,k)})}. \tag{5}$$

Therefore, we can calculate the resulting weight of the particle by

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i)})p(\mathbf{r}_t^{(i)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})}{\pi(\mathbf{r}_t^{(i)}|\mathbf{z}_{1:t},\mathbf{u}_{1:t-1},\mathbf{r}_{1:t-1}^{(i)})} \\ &= w_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i)})p(\mathbf{r}_t^{(i)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})}{p(\mathbf{r}_t^{(i,0)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})\frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i,1)})}{p(\mathbf{z}_t|\mathbf{r}_t^{(i,0)})}\frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i,2)})}{p(\mathbf{z}_t|\mathbf{r}_t^{(i,1)})}\cdots\frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i,K)})}{p(\mathbf{z}_t|\mathbf{r}_t^{(i,K-1)})}} \\ &= w_{t-1}^{(i)} \frac{p(\mathbf{z}_t|\mathbf{r}_t^{(i,0)})p(\mathbf{r}_t^{(i)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})}{p(\mathbf{r}_t^{(i,0)}|\mathbf{r}_{t-1}^{(i)},\mathbf{u}_{t-1})}. \end{aligned} \tag{6}$$

With this procedure, particles move toward the local optimum of likelihood distribution and finally forms a swarm following the target distribution as Fig. 1 (b). It can be thought of as a post processing after initial particle propagation.

The computational time to get one particle will increase when the PSO iteration increases, because we have to calculate the fitness (likelihood) of particles at every time they move. With PSO, however, since it is possible to keep track of the optimum values with a smaller number of particles than generic particle filters, the total computation time can be greatly reduced.

The necessary condition for applying PSO to the obtained particles is that the maximum likelihood point must be in the range of obtained particles. To meet this condition, we have to keep the range of particles to appropriate size. This can be achieved via repulsion dynamics between particles and between swarms, which will be explained in the next section.

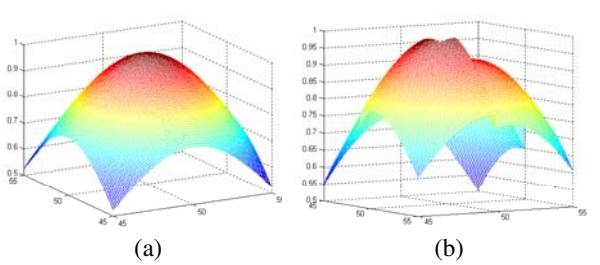
C. Exclusion between particles

To prevent a severe convergence of particles, we can put the repulsion forces between particles. Quantum Swarm Optimization (QSO) [13] introduce a concept of quantum particle to PSO. Quantum particle is non-moving particle, thus it does not get close to each other. Our swarms are composed of neutral particles and quantum particles. The neutral particles move according to the equation (1). The quantum particles have zero velocity and are distributed in wide area around a local optimum. This non-converged distribution of quantum particle provides a chance to search new region to the swarm. The position of quantum particle j is determined at the start of every PSO iteration by

$$\mathbf{r}_t^{(j,k)} \in B(\mathbf{g}_t, r_{cloud}), \tag{7}$$

where $B(\mathbf{g}_t, r_{cloud})$ represents a ball centered at $\mathbf{g}_t$ with a radius r_{cloud}.

When we move to the next particle filter step, we move the ball center $\mathbf{g}_t$ according to the motion prior $\mathbf{u}_t$ and form a new ball with a radius r_{cloud}. If some of these quantum particles have a higher likelihood than the current local optimum, then the neutral particles move toward these quantum particles and try to find better points. The size of r_{cloud} has to be set carefully because it has to cover the odometry error, but it cannot be too large. To apply the concept of QSO we use these quantum particles, but these are just assistants for the neutral (normal) particles. Therefore the quantum particles do not have their maps and do not perform the update of landmarks. When we calculate the likelihood of the quantum particles, we use the map of the best neutral particle among the swarm.



(a) (b)

Fig. 2. Shape of likelihood function on x and y coordinates of robot pose. (a) Identical data association for all particles. (b) Different data association from different robot poses.

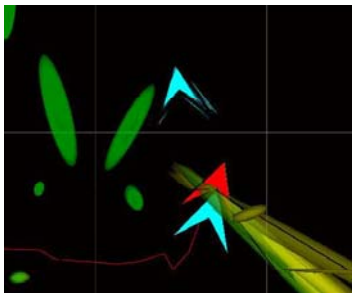


Fig. 3. Three swarms maintaining multiple modes. Red arrow: most probable robot pose, Cyan arrows: other particles.

D. Multiswarm

If the likelihood function has only one mode (local optimum), using one swarm is sufficient. However, if there is more than one mode, then it is very risky to keep only a single swarm. In a visual SLAM which uses visual landmarks like corner points, the result of data association severely changes the shape of likelihood function. If the data association is identical for all robot poses, then the shape of likelihood function is concave like Fig. 2 (a). The projected landmark position, however, is different according to the sampled robot pose, and the data association based on the projected landmark position can be different from particle to particle, which varies the shape of the likelihood function as shown in Fig. 2 (b). If we put the swarm at the mode with false robot pose and do not have swarm for the true robot pose, then the distance between these mode will be large and finally we will lose the true robot pose and will not be able to recover it.

To track multiple modes of the likelihood distribution, we use multiple swarms. Through particle filtering, however, these swarms may also get close to each other and be converged. Therefore, we need a mechanism to prevent the convergence of swarms as similar to the case of particles. This is achieved by separating close swarms each other. We set the distance threshold that determines whether two swarms are in the range of same mode as r_{excl}. After updating all particles in each swarm, we calculate the distance d_{nm} between each pair of swarms, given by the distance between the two swarms' best position $\mathbf{g}_n$ and $\mathbf{g}_m$, where n,m are the indices of the swarm. If d_{nm} is smaller than the threshold r_{excl}, then only one of the two swarms survives. Here, we select the swarm with higher likelihood from $\mathbf{g}_n$, $\mathbf{g}_m$ as a survivor. Not survived swarm is then re-initialized or removed, according to the swarm birth and death strategy in [17].

IV. Proposed SLAM system

The proposed particle filter can be applied to any SLAM system, and can especially contribute to vision-based SLAM which uses visual landmark like corner point. There can be similar patterns in visual scenes which can cause some ambiguities in possible robot poses. Maintaining multiple swarms can remedy this problem. Among the vision-based SLAM systems, we apply the proposed particle filter to the ceiling vision based SLAM [18]. Ceiling scenes are easy to match due to their scale invariant property, but they suffer from the mistakes of data association caused by similar patterns in ceiling. When there are similar or repeated patterns, the robot has to keep track of as many likely trajectories as possible.

We define the distance in the problem space to apply the concept of multiswarm and exclusion of particles. The robot pose of particle i is given by $\mathbf{r}_t^{(i)} = [x_i, y_i, \theta_i]_t^T$, so the problem space is $\mathbb{R}^3$. We have to give a weight to each dimension because x_i, y_i and θ_i have different measures. In our system x_i and y_i are measured by *cm*, and angle θ_i is measured by *radian*, thus the compensation for different unit is required. As a result, we can define the measure for distance $d_r(\mathbf{r}^{(i)}, \mathbf{r}^{(j)})$ between robot pose $\mathbf{r}^{(i)}$ and $\mathbf{r}^{(j)}$ as equation:

$$d_r(\mathbf{r}_t^{(i)}, \mathbf{r}_t^{(j)}) = (x_i - x_j)^2 + (y_i - y_j)^2 + a^2 Trunc^2(\theta_i - \theta_j), \quad (8)$$

where a is a weight value for angle, we set it 100 in our experiment. r_{cloud} and r_{excl} all use this measurement. $Trunc(\theta)$ is the truncation function that makes θ always between $-\pi$ and π.

The weight of a particle is calculated from the average distance between the projected positions of the 3D landmarks to an image and the detected points of landmarks in the image. The smaller distance means more probable particle of robot pose, so by using exponential model we can define the likelihood of the particle by

$$p(\mathbf{z}|\mathbf{r}_t^{(i)}) = \exp\left[-\frac{1}{N}\sum_{l=1}^{N} d(\mathbf{z}_l, \hat{\mathbf{z}}_l)\right], \quad (9)$$

where N is the number of observed landmarks in current scene. $\hat{\mathbf{z}}_l$ and $\mathbf{z}_l$ are the projected position and the measured position of the landmark $\mathbf{L}^l$ in the image plane, respectively. $d(\mathbf{z}_l, \hat{\mathbf{z}}_l)$ is the Euclidean distance between them.

V. Experiment

To evaluate the performance of the proposed algorithm, we perform two sets of experiments. One is an experiment with a real ceiling vision based SLAM system, and the other is an experiment on synthetic simulation. Since the ground truth of the robot trajectory and the landmark positions are not available for real experiment, we cannot perform an error analysis of the system. Instead, we test the algorithm indirectly with real system by comparing the maximum likelihood and the effective number particles. We perform a quantitative error analysis using synthetic simulation which has the same model of the real system. We compare our algorithm with the generic particle filter, the linearized optimal importance method [6] used in FastSLAM-2[5].

A. Real System Experiment

We perform the experiment in an indoor environment because the proposed algorithm uses ceiling vision. The robot moves on a planar floor and observes the landmarks on the ceiling or wall through a wide FOV (field of view) camera that is pointing in the upward direction.

Fig. 4 (a) and (b) show the resulting maps obtained by the generic particle filter and our proposed particle filter, respectively. The positions and uncertainties of landmarks are indicated by green ellipsoids. Although we cannot evaluate the accuracy of the result since we do not have a ground truth, qualitative comparison is possible using knowledge of the structure of experiment environment. In Fig. 4, the environment is composed of the wall, door and ceiling, thus we can see the quality of the mapping by checking the co-planarity of landmarks in the map. We can see that the map of proposed particle filter is better than that of generic particle filter in that sense.

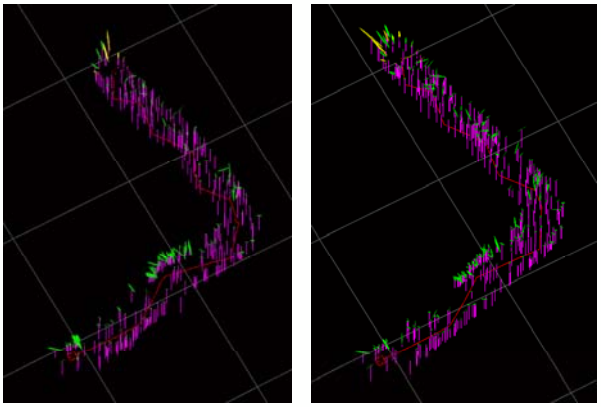


Fig. 4. Mapping results with real system: (a) Generic PF (b) Proposed PF

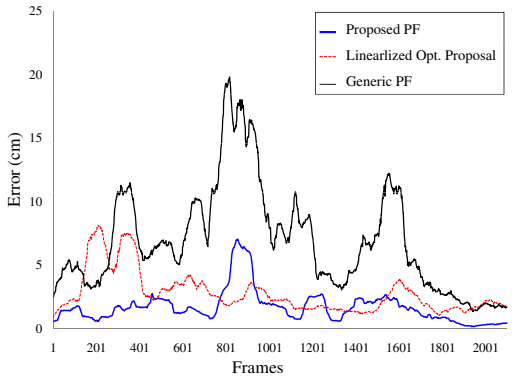


Fig. 6. Synthetic data simulations, with the same computation time for all algorithms: The plot of pose error between the ground truth and the best sample position.

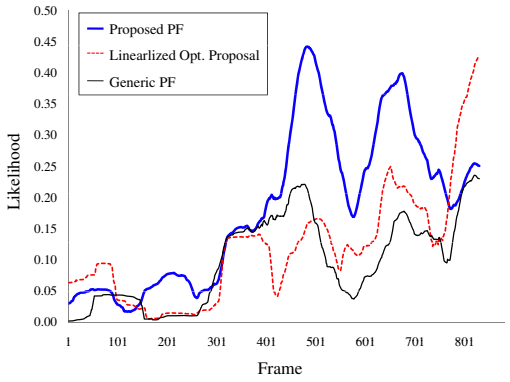


Fig. 5. The plot of the maximum likelihood found by particle filters.

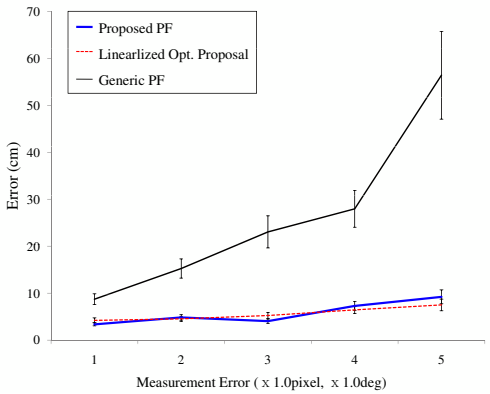


Fig. 7. The plot of pose error with varying measurement noise.

We give different number of particles to the filters because each filter needs different time for dealing with one particle. The generic particle filter uses 300 particles while the linearized optimal proposal method uses 180 particles. The proposed particle filter has 200 particles which are divided into four swarms and each swarm has 25 neutral particles and 25 quantum particles which perform three PSO iterations for one frame. This compensation makes the elapsed time for each algorithm almost the same.

The mapping result of the linearized optimal proposal method is almost the same with the result of our method. Even though we cannot plot the error of robot pose, we can plot the maximum likelihood found by each particle filter. This is helpful for comparing the sampling performance of the filters. In Fig. 5, the proposed particle filter has higher maximum likelihoods than the other filters in more regions. The average likelihoods are 0.178 (proposed PF), 0.129 (Linearized optimal proposal), and 0.0996 (Generic PF) for each filter. This result is gained from the optimization ability of the original PSO algorithm. The proposed particle filter does not generate particles at once; it adjusts the particles after particle transition by motion model. This enables the particles to find a better position with respect to the likelihood. If the measurement error is small, then our filter will give promising results. If not, the particle distribution will be overfit to the likelihood and the filter performance can be degraded. This is experimented in synthetic data simulation.

B. Synthetic Simulation

To evaluate the accuracy of each particle filter quantitatively, we test each filter to the synthetic simulation of the ceiling vision based SLAM system. To compare only filtering accuracy, we set all algorithms to the same odometry, same real motion (Gaussian error and abrupt motions are added to the odometry) and same data association results. The robot moves total 2037*cm* trajectory. Again, we tested three algorithms, the generic particle filter, the linearized optimal importance method, and our proposed algorithm. We tested the algorithms under two different conditions: same number of particles and same computation time. For all results, a position error for robot contained only translation error for a convenience.

TABLE I

EXPERIMENTS WITH SAME NUMBER OF PARTICLES

	Generic PF	Opt. Proposal	Proposed PF
# particle	30	30	30
Time (sec)	27.02	38.25	32.01
Avg. Error (cm)	11.16	2.662	2.699
Stdev. (cm)	1.80	0.385	0.286

TABLE II

EXPERIMENTS WITH SAME COMPUTATION TIME

	Generic PF	Opt. Proposal	Proposed PF
# particle	55	30	43
Time (sec)	39.49	38.25	38.28
Avg. Error (cm)	6.612	2.662	1.606
Stdev. (cm)	0.987	0.385	0.244

With the experiment of same particle number condition, we can compare the quality of generated particles. Table I summarizes the results. This experiment is performed with a measurement error of 1.0pixel and 1.0degree of the measurement. With the same number of particles, the proposed algorithm and linearized optimal proposal shows nearly equal results in average position errors. The proposed algorithm gives the least standard deviation of error, which means it shows a stable performance. Since the proposed algorithm requires less computation time for generating one particle than the linearized optimal proposal method, the performance of the proposed algorithm is improved when we use the same computation time for all filters. The summary of same computation time experiment is Table II.

Fig. 6 shows the plot of position error during the robot's travel. Initially, all the filters show similar patterns but as they move the difference is increased. We simulate the large odometry noises induced by external force to the robot, and that time the plots change severely. For the generic particle filter, it is hard to recover the correct trajectory after this motion, while the other filters quickly reduce the error.

We also tested the performance of each filter with a varying measurement noise. Since the proposed method and the linearized optimal proposal method generate particles using recent observation, these algorithms cannot give good results when the quality of the observation is poor. Fig. 7 shows the result on this experiment. As we can see, the proposed algorithm shows better results on low measurement noise and worse results on high measurement noise than the linearized optimal proposal method. Actually, 5 pixels and 5 degrees are quite large errors in image based measurement, thus we can conclude that the proposed algorithm can give better results in general environment.

VI. CONCLUSION

In this study we have proposed a novel particle filtering method which can be used to resolve the problem of particle impoverishment by employing the Particle Swarm Optimization-based approaches. With this method we can perform efficient sampling in terms of the number of required particles by combining the generic particle filter with the PSO which is equipped with the concept of quantum particles and multiple swarms. Our algorithm can be applied to not only the landmark-based SLAM, but also to grid-based SLAM, wherever we can evaluate a likelihood from observation. Moreover, the proposed particle filter can be applied not only to SLAM, but also to any kind of application which uses the particle filter-based approaches such as object tracking.

VII. ACKNOWLEDGMENTS

This research was supported in part by SAMSUNG THALES, and in part by the ITRC program of MKE/IITA through 3DRC (IITA-2008- C1090-0801-0018), Korea.

REFERENCES

[1] K. Murphy, "Bayesian map learning in dynamic environments," in *Proc. Advances in Neural Information Processing Systems*, Colorado, USA, 1999.

[2] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proc. Conference on Uncertainty in Artificial Intelligence*, 2000.

[3] D. Fox, "Kld-sampling: Adaptive particle filters," in *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, 2001.

[4] A. Soto, "Self adaptive particle filter," in *Proc. International Joint Conferences on Artificial Intelligence*, Edinburgh, UK, 2005.

[5] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proc. National Conference on Artificial Intelligence*, 2002.

[6] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering."

[7] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," in *Proc. National Conference on Artificial Intelligence*, 2002.

[8] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco, USA: Morgan Kaufmann, 2001.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE International Conference on Neural Networks*, Perth, Australia, 1995.

[10] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, no. 1, pp. 29–41, 1996.

[11] D. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees algorithm - a novel tool for complex optimisation problems," in *Proc. Innovative Production Machines and Systems Conference*, 2006.

[12] A. Carlisle and G. Dozier, "Adapting particle swarm optimization to dynamic environments," in *Proc. International Conference on Artificial Intelligence*, Las Vegas, USA, 2000.

[13] T. Blackwell and J. Branke, "Multiswarms, exclusion and anti-convergence in dynamic environments," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 459–472, 2006.

[14] X. Zhang, W. Hu, S. Maybank, X. Li, and M. Zhu, "Sequential particle swarm optimization for visual tracking," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, Anchorage, USA, 2008.

[15] G. Tong, Z. Fang, and X. Xu, "A particle swarm optimized particle filter for nonlinear system state estimation," in *Proc. IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.

[16] Q. Wang, L. Xie, J. Liu, and Z. Xiang, "Enhancing particle swarm optimization based particle filter tracker," *Lecture Notes in Computer Science*, vol. 4114, pp. 1216–1221, 2006.

[17] T. Blackwell, "Particle swarm optimization in dynamic environments," *Evolutionary Computation in Dynamic and Uncertain Environments*, pp. 29–49, 2007.

[18] W. Jeong and K. Lee, "Cv-slam: A new ceiling vision-based slam technique," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, China, 2005.