A Square Root Unscented FastSLAM With Improved Proposal Distribution and Resampling

Ramazan Havangi, Hamid D. Taghirad, *Senior Member, IEEE*, Mohammad Ali Nekoui, and Mohammad Teshnehlab

***Abstract*—An improved square root unscented fast simultaneous localization and mapping (FastSLAM) is proposed in this paper. The proposed method propagates and updates the square root of the state covariance directly in Cholesky decomposition form. Since the choice of the proposal distribution and that of the resampling method are the most critical issues to ensure the performance of the algorithm, its optimization is considered by improving the sampling and resampling steps. For this purpose, particle swarm optimization (PSO) is used to optimize the proposal distribution. PSO causes the particle set to tend to the high probability region of the posterior before the weights are updated; thereby, the impoverishment of particles can be overcome. Moreover, a new resampling algorithm is presented to improve the resampling step. The new resampling algorithm can conquer the defects of the resampling algorithm and solve the degeneracy and sample impoverishment problem simultaneously. Compared to unscented FastSLAM (UFastSLAM), the proposed algorithm can maintain the diversity of particles and consequently avoid inconsistency for longer time periods, and furthermore, it can improve the estimation accuracy compared to UFastSLAM. These advantages are verified by simulations and experimental tests for benchmark environments.**

***Index Terms*—Particle swarm optimization (PSO), simultaneous localization and mapping (SLAM), square root unscented Kalman filter (SRUKF), unscented FastSLAM (UFastSLAM).**

I. Introduction

SIMULTANEOUS localization and mapping (SLAM) is a key issue to achieve intelligent navigation for mobile robots. Two main computational solutions to SLAM are the extended Kalman filter based SLAM (EKF-SLAM) and FastSLAM [1]. However, EKF-SLAM has two major problems, namely, the computational complexity and data association [2], [3]. The FastSLAM approach has been proposed as an alternative approach to solve the aforementioned problem. In FastSLAM, the particle filter is used for the mobile robot position estimation, and EKF is used for the feature location estimation [4], [5].

There have been many investigations on FastSLAM [6], [7]. However, FastSLAM also suffers from a number of drawbacks, namely, the problem caused by the derivation of the Jacobian matrices and the linear approximations of the nonlinear functions [8], [9]. To solve these problems, many authors have proposed UFastSLAM [8]–[11]. UFastSLAM overcomes the drawbacks caused by linearization in the FastSLAM framework. In UFastSLAM, the linearization process with Jacobian calculations is removed by applying the unscented transformation (UT) [12], [13]. In [11], a full version of the UFastSLAM algorithm is presented. In this approach, unscented Kalman filter (UKF) is used to update the mean and covariance of the feature and to initialize new features. Also, UT is used in the prediction step of the vehicle state, and the unscented particle filter provides a better proposal distribution without the accumulation of linearization errors and without the need to calculate the Jacobian matrices in the measurement updates [11].

As the vehicle and the feature states are estimated without accumulating linearization errors in UFastSLAM, the accuracy of the state estimation has been significantly improved. However, one of the most costly operations in UFastSLAM is the calculation of the square root of the state variable covariance matrix at each time. From the consistency viewpoint, this approach is performing more consistently for longer time periods compared to the other methods. However, since, in UFastSLAM, residual systematic resampling (RSR) is used as the resampling method, this usually causes the loss of diversity of particles and consequently threatens the consistency. To overcome these problems, in this paper, a new machinery called square root unscented FastSLAM (SRUFastSLAM) is proposed in order to reduce the computational cost and increase accuracy and consistency. SRUFastSLAM uses the square root unscented Kalman filter (SRUKF) to generate the importance density, initialization, and estimation of features. Aside from the merit of reducing the computational cost, SRUFastSLAM has some other advantages such as increasing consistency which leads to more numerical stability and better performance. This is mainly because of the fact that all resulting covariance matrices are guaranteed to remain positive semidefinite. Furthermore, as the optimal choice of the proposal distribution and that of the resampling method are crucial to improve accuracy and consistency, SRUFastSLAM is optimized by improving the sampling and resampling steps.

To optimize the sampling step, particle swarm optimization (PSO) has been merged into the importance sampling step of SRUFastSLAM. By this means, the particle set effectively tends to the high probability posterior region before resampling, and therefore, the distribution of the samples is significantly improved.

In order to reduce the impact of resampling on the accuracy and consistency of SRUFastSLAM, a new resampling approach

Manuscript received May 24, 2011; revised April 26, 2012, August 16, 2012, and November 22, 2012; accepted May 26, 2013. Date of publication July 3, 2013; date of current version October 18, 2013.

The authors are with the Faculty of Electrical Engineering, Systems and Control Department, K. N. Toosi University of Technology, Tehran 19697, Iran (e-mail: Havangi@ee.kntu.ac.ir; Taghirad@eetd.kntu.ac.ir; Manekoui@eetd.kntu.ac.ir; Teshnehlab@eetd.kntu.ac.ir).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2013.2270211

is also proposed. The new resampling algorithm can maintain the diversity of particles and ensure that the resampled particles asymptotically approximate the samples from the posterior probability density function (pdf) of the true state. The proposed resampling method is computationally inexpensive since it is conducted only on a part of the particles. Simulation results show that the improvement of the proposed algorithm over the resampling algorithm is manifested in three aspects: increased estimation accuracy, reduced number of particles required to achieve the same level of accuracy, and maintained diversity of particles.

The rest of this paper is organized as follows. In Section II, the SLAM problem and the required background are reviewed. The SRUFastSLAM framework and its theoretical principles are presented in Section III. The improved SRUFastSLAM is presented in Section IV. In Section V, the results are shown, and the concluding remarks are given in Section VI.

II. Background

A. SLAM Problem

From a probabilistic point of view, the SLAM problem involves the estimation of the posterior over the robot path together with the map [5]

$$p(s^t, \Theta | z^t, u^t, n^t) \tag{1}$$

in which $s^t = \{s_1, \ldots, s_t\}$ is the robot path, s_t is the robot pose at time t, Θ is the map, and $z^t = \{z_1, \ldots, z_t\}$ and $u^t = \{u_1, \ldots, u_t\}$ are the measurements and controls up to time t, respectively. The map Θ consists of a collection of features, each of which will be denoted by θ_n, and the total number of stationary features will be denoted by N. The FastSLAM algorithm is developed based on the following factorization [5]:

$$p(s^t, \Theta | z^t, u^t, n^t) = p(s^t | z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n | s^t, z^t, u^t, n^t) \tag{2}$$

where $n^t = \{n_1, \ldots, n_t\}$ is the data association, in which each n_t specifies the identity of the landmark observed at time t. Each particle m in FastSLAM is denoted by [5]

$$S_t^{[m]} = \left\{ s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \ldots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\} \tag{3}$$

where $[m]$ indicates the index of the particle, $s^{t,[m]}$ is the m-th particle's path estimate, and $\mu_{N,t}^{[m]}$ and $\Sigma_{N,t}^{[m]}$ are the mean and the covariance of the Gaussian distribution representing the N-th feature location conditioned on the robot path. In FastSLAM, the robot pose s_t is sampled as follows [5]:

$$s_t \sim q(s_t | s^{t-1,[m]}, z^t, u^t, n^t) \tag{4}$$

and the importance weight $w_t^{[m]}$ is given by

$$\begin{aligned} w_t^{[m]} &= \frac{\text{target distribution}}{\mathit{proposal}\ \text{distribution}} \\ &= \frac{p(s^{t,[m]} | z^t, u^t, n^t)}{q(s^{t-1,[m]} | z^{t-1}, u^{t-1}, n^{t-1}) q(s_t | s^{t-1,[m]}, z^t, u^t, n^t)}. \end{aligned} \tag{5}$$

In the final step of FastSLAM, particles are resampled according to their weights.

B. PSO

PSO is a population-based search algorithm based on the simulation of the social behavior of birds within a flock [14]. PSO is initialized with a group of random particles and then searches for optima by updating generations. Suppose that the search space dimension is D and the number of particles is NP, the position and velocity of the ith particle are represented by $x_i = [x_{i1}, \ldots, x_{iD}]$ and $v_i = [v_{i1}, \ldots, v_{iD}]$, respectively. Let $P_{bi} = [p_{i1}, \ldots, P_{iD}]$ denote the best position that particle i has achieved so far, and P_g denote the best position of P_{bi} for any $i = 1, \ldots, NP$. The PSO algorithm may be performed by the following formulations [14]:

$$\begin{aligned} v_i(k) = {} & w v_i(k-1) + c_1 r_1 \left(P_{bi} - x_i(k-1)\right) \\ & + c_2 r_2 \left(P_g - x_i(k-1)\right) \end{aligned} \tag{6}$$

$$x_i(k) = x_i(k-1) + v_i(k) \tag{7}$$

where k represents the iteration number and c_1 and c_2 are some positive coefficients. Usually, $c_1 = c_2 = 2$, r_1 and r_2 are random numbers in the interval (0, 1), and w is the inertial weight. Note that PSO has strong global search ability if large inertial weight w is used.

III. SRUFastSLAM

SRUFastSLAM is based on the UT. The proposed algorithm consists of the sampling strategy, feature update, calculation of the importance weight, and resampling.

A. Sampling Strategy

The choice of proposal distributions is very important in the design of FastSLAM. The most common strategy is to sample the transition motion (or prior distribution) [5]. However, this strategy may fail when the prior distribution is much broader than the likelihood or most measurements that appear in the tail of the proposal distribution. Several researchers have introduced the most current observations into the proposal distribution [5]. To overcome the drawbacks caused by linearization, UKF is used in the design of the proposal distribution in UFastSLAM [10], [11]. However, the calculation of the square root of the state variable covariance matrix at each time is one of the computationally intensive operations in UFastSLAM.

To reduce computational cost and to increase consistency, in SRUFastSLAM, the mean and covariance of the vehicle are computed by applying the square root unscented particle filter (SRUPF) technique [15]. In this algorithm, poses are sampled from a proposal distribution as (4). SRUFastSLAM generates posterior probability with SRUPF. In this approach, each particle is updated at the measurement time using SRUKF. The first step for implementing SRUFastSLAM is to form an augmented state and augmented covariance matrix by appending the mean and covariance of the process noise vector. Assuming that the mean of process noise is zero, then the mean and covariance of the augmented state are respectively given as follows:

$$s^{a[m]}_{t-1|t-1} = \begin{bmatrix} s^{[m]}_{t-1|t-1} \\ 0 \end{bmatrix}, \quad P^{a[m]}_{t-1|t-1} = \begin{bmatrix} P^{[m]}_{t-1|t-1} & 0 \\ 0 & Q_t \end{bmatrix} \tag{8}$$

where $s^{a[m]}_{t-1|t-1}$ is the augmented vector, Q_t is the process noise covariance, and $s^{[m]}_{t-1|t-1}$ and $P^{[m]}_{t-1|t-1}$ are the previous mean and covariance of the robot pose, respectively. Use the Cholesky factor of $P^{a[m]}_{t-1|t-1}$ as follows:

$$S^{a[m]}_{t-1|t-1} = chol \begin{bmatrix} P^{[m]}_{t-1|t-1} & 0 \\ 0 & Q_t \end{bmatrix}. \tag{9}$$

Since Cholesky factorization decomposes a positive-definite matrix $P^{a[m]}_{t-1|t-1}$ into the product of an upper triangular matrix and its transpose, we can write $P^{a[m]}_{t-1|t-1} = S^{a[m]}_{t-1|t-1} S^{a[m]'}_{t-1|t-1}$. To predict over a time step $t-1$ to t, the first sigma point $\delta^{a[[0]][m]}_{t-1|t-1}$ is defined to be equal to the current state, and further $2n$ sigma points are determined by adding and subtracting the ith column of $S^{a[m]}_{t-1|t-1}$ to the current mean. Therefore, a symmetric set of $2n+1$ sigma points $\delta^{a[i][m]}_{t-1|t-1}$ for the augmented state will appear as follows:

$$\begin{aligned}
\delta^{a[0][m]}_{t-1|t-1} &= s^{a[m]}_{t-1|t-1} \\
\delta^{a[i][m]}_{t-1|t-1} &= s^{a[m]}_{t-1|t-1} + \gamma \left(S^{a[m]}_{t-1|t-1} \right)_i, \qquad i = 1, \dots n \\
\delta^{a[i][m]}_{t-1|t-1} &= s^{a[m]}_{t-1|t-1} - \gamma \left(S^{a[m]}_{t-1|t-1} \right)_i, \qquad i = n+1, \dots, 2n
\end{aligned} \tag{10}$$

where γ is $\gamma = \sqrt{n + \lambda}$ in which λ is defined as $\lambda = n(\alpha^2 - 1)$ and $(S_{t-1|t-t})_i$ is the ith column of $S^{a[m]}_{t-1|t-1}$. The constant α controls the size of the sigma-point distribution, and it is usually set to a small positive value (e.g, $10^{-14} \leq \alpha \leq 1$). Each sigma point $\delta^{a[i][m]}_{t-1|t-1}$ contains the state and control noise components given by

$$\delta^{a[i][m]}_{t-1|t-1} = \begin{bmatrix} \delta^{[i][m]}_{t-1|-1} & \delta^{u[i][m]}_{t} \end{bmatrix}^T. \tag{11}$$

The set of sigma points $\delta^{a[i][m]}_{t-1|t-1}$ is transformed through the motion model as

$$\overline{s}^{[i][m]}_{t|t-1} = f_v \left(\delta^{[i][m]}_{t-1|t-1}, u_t + \delta^{u[i][m]}_t \right). \tag{12}$$

Here, f_v is the nonlinear motion function and $\overline{s}^{[i][m]}_{t|t-1}$ is the transformed sigma point of the vehicle state. The predicted mean is calculated from the transformed sigma points as follows:

$$s^{[m]}_{t|t-1} = \sum_{i=0}^{2n} \omega^{[i]}_m \overline{s}^{[i][m]}_{t|t-1} \tag{13}$$

where a weight $\omega^{[i]}_m$ is calculated as follows:

$$\omega^{[0]}_m = \frac{\lambda}{(n+\lambda)}, \omega^{[i]}_m = \frac{\lambda}{2(n+\lambda)}, \quad i = 1, \dots, 2\text{n}. \tag{14}$$

Calculating the Cholesky factorization is computationally intensive, and it involves the computation of a set of weighted deviations as

$$\begin{aligned}
e_i &= \sqrt{\omega^{[0]}_c} \left(\overline{s}^{[i][m]}_{t|t-1} - s^{[m]}_{t|t-1} \right), \qquad i = 1, \dots, 2\text{n} \\
e_0 &= \sqrt{\omega^{[1]}_c} \left(\overline{s}^{[0][m]}_{t|t-1} - s^{[m]}_{t|t-1} \right)
\end{aligned} \tag{15}$$

and performing QR decomposition on the matrix $B = [e_1 \dots e_{2n}]$ as follows:

$$S^{[m]}_{t|t-1} = \text{qr} \left\{ \left[\sqrt{\omega^{[1]}_c} \left(\overline{s}^{[1:2n][m]}_{t|t-1} - s^{[m]}_{t|t-1} \right) \right] \right\} \tag{16}$$

where weight $\omega^{[i]}_c$ is calculated as follows:

$$\begin{aligned}
\omega^{[0]}_c &= \frac{\lambda}{(n+\lambda)} + (1 - \alpha^2 + \beta), \omega^{[i]}_c \\
&= \frac{\lambda}{2(n+\lambda)}, \qquad i = 1, \dots, 2n.
\end{aligned} \tag{17}$$

The parameter β is a nonnegative scalar which can be used to incorporate the prior knowledge of the distribution. For any Gaussian distribution, $\beta = 2$ is optimal. The predicted factor $S^{[m]}_{t|t-1}$ is then found using the Cholesky updating as

$$S^{[m]}_{t|t-1} = \text{cholupdate} \left\{ S^{[m]}_{t|t-1}, \left(\overline{s}^{[0][m]}_{t|t-1} - s^{[m]}_{t|t-1} \right), \omega^{[0]}_c \right\}. \tag{18}$$

The function cholupdate efficiently transforms the Cholesky decomposition of a matrix A into the Cholesky decomposition of the matrix $A + XX^T$, where X is a column vector. When some features are observed, data association provides their identities, and hence, predicted measurement $\overline{z}^{[m]}_t$ is calculated as follows:

$$\begin{aligned}
\zeta^{[i][m]}_t &= h \left(\overline{s}^{[i][m]}_{t|t-1}, \mu^{[m]}_{n,t-1} \right) \\
\overline{z}^{[m]}_t &= \sum_{i=0}^{2n} w^{[i]}_m \zeta^{[i][m]}_t \\
S_{z_t} &= qr \left\{ \left[w^{[1]}_c \left(\zeta^{[1:2n][m]}_t - \overline{z}^{[m]}_t \right), \sqrt{R_t} \right] \right\} \\
S_{z_t} &= \text{cholupdate} \left\{ S_{z_t}, \left(\zeta^{[0][m]}_t - \overline{z}^{[m]}_t \right), w^{[0]}_c \right\}.
\end{aligned} \tag{19}$$

Here, S_{z_t} is square and triangular, and $h(.)$ is the observation model. In order to determine how much the predicted mean and square root of the covariance are adjusted based on the measurements, the gain matrix $K^{[m]}_t$ is calculated

$$K^{[m]}_t = \left(P^{[m]}_{\delta v} / S^T_{z_k} \right) \Big/ S_{z_k} \tag{20}$$

where cross-covariance $P^{[m]}_{\delta v}$ is as follows:

$$P^{[m]}_{\delta v} = \sum_{i=0}^{2n} \omega^{[i]}_c \left(\overline{s}^{[i][m]}_{t|t-1} - s^{[m]}_{t|t-1} \right) \left(\zeta^{[i][m]}_t - \overline{z}^{[m]}_t \right)^T. \tag{21}$$

Finally, the state mean $s_{t|t}^{[m]}$ and square root of the covariance matrix $S_{t|t}^{[m]}$ are updated as

$$\begin{aligned} s_{t|t}^{[m]} &= s_{t|t-1}^{[m]} + K_t^{[m]}(z_t - \overline{z}_t^{[m]}) \quad (22)\\ U &= K_t^{[m]} S_{z_t}\\ S_{t|t}^{[m]} &= \text{cholupdate}\left\{S_{t|t-1}^{[m]}, U, -1\right\}. \quad (23)\end{aligned}$$

From the Gaussian distribution generated by the estimated mean and covariance of the vehicle, the state of each particle is sampled as

$$s_t^{[m]} \sim N\left(s_t; s_{t|t}^{[m]}, P_{t|t}^{[m]}\right) \quad (24)$$

where the covariance matrix of the robot pose is as follows:

$$P_{t|t}^{[m]} = \left(S_{t|t}^{[m]}\right)^T S_{t|t}^{[m]}. \quad (25)$$

B. Feature Update

Feature update is applied if a landmark n is observed at time t. If the feature is not observed $(n \neq n_t)$, the posterior landmark remains unchanged. For the observed feature $(n = n_t)$, the update is specified through the following equation:

$$\begin{aligned} &p(\theta_{n_t}|s^{t,[m]}), n^t, z^t)\\ &= \eta \underbrace{p(z_t|\theta_{n_t}, s^{t,[m]}, n^t, z^{t-1})}_{\sim N\left(z_t, h\left(\theta_{n_t}, s_{t|t}^{[m]}\right), R_t\right)} \underbrace{p(\theta_{n_t}|s^{t,[m]}, n^t, z^{t-1})}_{\sim N\left(\theta_{n_t}, \mu_{n,t-1}^{[m]}, \Omega_{n,t-1}^{[m]T}\Omega_{n,t-1}^{[m]}\right)} \end{aligned} \quad (26)$$

where R_t is the measurement noise covariance. The probability $p(\theta_{n_t}|s^{t,[m]}, n^t, z^{t-1})$ at time $t-1$ is represented by a Gaussian distribution with mean $\mu_{n,t-}^{[m]}$ and covariance $\Sigma_{n_t,t-1}^{[m]} = (\Omega_{n_t,t-1}^{[m]})^T \Omega_{n_t,t-1}^{[m]}$ in which $\Omega_{n_t,t-1}^{[m]}$ is the feature square root of the covariance matrix. In SRUFastSLAM, UT is used to approximate $h(\theta_{n_t}, s_t^{[m]})$. For this purpose, the sigma points are defined using the previously registered mean and square root of the covariance matrix as follows:

$$\begin{aligned} \chi^{[0][m]} &= \mu_{n_t,t-1}^{[m]}\\ \chi^{[i][m]} &= \mu_{n_t,t-1}^{[m]} + \gamma\left(\Omega_{n_t,t-1}^{[m]}\right)_i, i = 1,\ldots,n\\ \chi^{[i][m]} &= \mu_{n_t,t-1}^{[m]} - \left(\gamma\Omega_{n_t,t-1}^{[m]}\right)_i i = n+1,\ldots,2\text{n}\\ \Sigma_{n_t,t-1}^{[m]} &= \left(\Omega_{n_t,t-1}^{[m]}\right)^T \Omega_{n_t,t-1}^{[m]}. \end{aligned} \quad (27)$$

The sigma points are propagated through the measurement model as follows:

$$\begin{aligned} \overline{Z}_t^{[i][m]} &= h\left(\chi^{[i][m]}, s_{t|t}^{[m]}\right), i = 0,\ldots,2n\\ \hat{z}_t^{[m]} &= \sum_{i=0}^{2n} \omega_g^{[i]} \overline{Z}_t^{[i][m]}\\ \overline{S}_{ty}^{[m]} &= qr\left\{\left[\omega_c^{[1]}\left(\overline{Z}_t^{[i][m]} - \hat{z}_t^{[m]}\right), \sqrt{R_t}\right]\right\}\\ \overline{S}_{ty}^{[m]} &= \text{cholupdate}\left\{\overline{S}_{ty}^{[m]}, \left(\overline{Z}_t^{[i][m]} - \hat{z}_t^{[m]}\right), \omega_c^{[0]}\right\} \end{aligned} \quad (28)$$

where $s_{t|t}^{[m]}$ is the current vehicle state of the mth particle. The cross-covariance between the state and observation $\bar{\Sigma}_{t\chi Z}^{[m]}$ and the gain of filter $\bar{K}_t^{[m]}$ are determined as follows:

$$\begin{aligned} \overline{\Sigma}_{t\chi Z}^{[m]} &= \sum_{i=0}^{2n} \omega_c^{[i]}\left(\chi^{[i][m]} - \mu_{n_t,t-1}^{[m]}\right)\left(\overline{Z}_t^{[i][m]} - \hat{z}_t^{[m]}\right)^T\\ \overline{K}_t^{[m]} &= \left(\overline{\Sigma}_{t\chi Z}^{[m]} / \left(\overline{S}_{ty}^{[m]}\right)^T\right) / \overline{S}_{ty}^{[m]}. \end{aligned} \quad (29)$$

Using this gain, the mean $\mu_{n_t,t}^{[m]}$ and the square root of covariance $\Omega_{n,t}^{[m]}$ of the nth feature are updated as

$$\begin{aligned} \mu_{n_t,t}^{[m]} &= \mu_{n_t,t-1}^{[m]} + \overline{K}_t^{[m]}\left(z_t - \hat{z}_t^{[m]}\right)\\ U &= K_t^{[m]} \overline{\Sigma}_{t\chi Z}^{[m]}\\ \Omega_{n,t}^{[m]} &= \text{cholupdate}\left\{\Omega_{n,t-1}^{[m]}, U, -1\right\}. \end{aligned} \quad (30)$$

C. Calculating Importance Weight and Resampling

The importance weight is determined as follows:

$$w_t^{[m]} = w_{t-1}^{[m]} \frac{p\left(z_t|s_t^{[m]}\right) p\left(s_t^{[m]}|s_{t-1}^{[m]}, u_t\right)}{N\left(s_{t|t}; s_{t|t}^{[m]}, P_{t|t}^{[m]}\right)}. \quad (31)$$

Since the variance of the importance weights increases over time, particles are resampled according to the weights [16].

IV. Optimization of SRUFastSLAM

In this section, the optimization of SRUFastSLAM is considered. As the performance of SRUFastSLAM seriously depends on the sampling strategy and resampling method, this optimization is considered with improving sampling and resampling steps.

A. Improved Sampling of SRUFastSLAM

To obtain better importance sampling distribution functions, in SRUFastSLAM, the proposal distribution is generated by SRUPF. It is valid when the posterior distribution can be closely approximated by a Gaussian distribution. However, the number of sigma points is small and may not represent complicated distributions adequately. To solve this problem, PSO is merged in SRUPF to move particles toward a region of the state space where a posterior probability density is significant before sampling. For this purpose, we consider the maximization posterior probability density as the cost function

$$\arg\max_s p(s^t|z^t, u^t, n^t). \quad (32)$$

On the other hand, by applying Bayes, the posterior over the robot path is given by

$$\begin{aligned} p(s^t|z^t, u^t, n^t) &= \eta p(z_t|s^t, z^{t-1}, u^t, n^t)\\ &\quad p(s_t|s^{t-1}, z^{t-1}, u^t, n^t) p(s^{t-1}|z^{t-1}, u^{t-1}, n^{t-1}) \end{aligned} \quad (33)$$

where η is a normalizing constant. As a result, the cost function is as follows:

$$\arg\max_s p(z_t|s^t, z^{t-1}, u^t, n^t)p(s_t|s^{t-1}, z^{t-1}, u^t, n^t) p(s^{t-1}|z^{t-1}, u^{t-1}, n^{t-1}). \quad (34)$$

This cost function is equal to

$$\arg\max_s \prod_{i=1}^{t} p(z_i|s^i, z^{i-1}, u^i, n^i) \prod_{i=1}^{t} p(s_i|s^{i-1}, z^{i-1}, u^i, n^i)p_0 \quad (35)$$

where p_0 is as

$$p_0 = p(s^{t-1}|z^{t-1}, u^{t-1}, n^{t-1}). \quad (36)$$

Therefore, the problem of moving particles toward a region of high probability can be easily stated as an optimization problem in the presence of constraints in terms of the dynamical and observation models as follows:

$$\arg\max_s \prod_{i=1}^{t} p(z_i|s^i, z^{i-1}, u^i, n^i) \prod_{i=1}^{t} p(s_i|s^{i-1}, z^{i-1}, u^i, n^i)p_0 \quad (37)$$

subject to

$$\begin{aligned} s_t &= f(s_{t-1}, u_t) + w_t \\ z_t &= h(s_t) + v_t. \end{aligned} \quad (38)$$

The objective function (37) can be reformulated in an equivalent and convenient form by taking logarithms as

$$\sum_{i=1}^{t} \log p(z_i|s^i, z^{i-1}, u^i, n^i) + \sum_{i=1}^{t} \log p(s_i|s^{i-1}, z^{i-1}, u^i, n^i) + \log p_0 = \sum_{i=1}^{t} \log p(z_i|s_i) + \sum_{i=1}^{t} \log p(s_i|s_{i-1}, u_{i-1}) + \log p_0. \quad (39)$$

If we consider the objective function as $f_0(s_t)$, it can be expressed recursively as follows:

$$f_o(s_t) = \log p(z_t|s_t) + \log p(s_t|s_{t-1}, u_{t-1}) + f_o(s_{t-1}) \quad (40)$$

where $f_0(s_{t-1})$ is as

$$f_o(s_{t-1}) = \sum_{i=1}^{t-1} \log p(z_i|s_i) + \sum_{i=1}^{t-1} \log p(s_i|s_{i-1}, u_{i-1}) + \log p_0. \quad (41)$$

The expression (40) contains the state transition probability density $p_{\omega_t}(s_i|s_{i-1}, u_{i-1})$ and the observation probability density $p_{v_t}(z_i|x_i)$ derived from the robot motion model and the observation model, respectively. By substituting the expressions of p_{v_t} and p_{ω_t} into the objective function, the problem of moving the particles to a high probability area can be reformulated as follows:

$$\mathrm{Min}\left(\log p_{\nu_t}(z_t|s_t) + p_{\omega_t}(s_t|s_{t-1}, u_{t-1})\right) \quad (42)$$

subject to

$$\begin{aligned} s_t &= f(s_{t-1}, u_t) + w_t \\ z_t &= h(s_t) + v_t. \end{aligned} \quad (43)$$

In general, this problem has no explicit analytical solution for nonlinear models. To solve this optimization problem, in this paper, an evolutionary computation procedure is used. Although there are many different types of evolutionary algorithms, we use PSO to solve the problem. This is because PSO finds the solution much faster than most of other evolutionary algorithms. In addition, it is easily implemented and has few parameter adjustments. To solve the optimization problem (42) and (43), we consider the fitness function of PSO as follows:

$$\mathrm{Fitness} = \log p_{\nu_t}(z_t|s_t) + p_{\omega_t}(s_t|s_{t-1}, u_{t-1}). \quad (44)$$

The particles should be moved such that the fitness function is optimal. This is done by tuning the position and velocity of the PSO algorithm. The standard PSO algorithm has some parameters that need to be determined before use. Most approaches use a uniform probability distribution to generate random numbers. However, it is difficult to obtain the fine tuning of the solution and escape from local minima using a uniform distribution. Hence, we use velocity updates based on the Gaussian distribution [17]. In this approach, there is no need to specify the coefficients c_1 and c_2. Furthermore, the inertial weight ω is set to zero, and an upper bound for the maximum velocity $v_{\max}$ is not necessary anymore [17]. Thus, the only parameter to be specified is the number of particles. The PSO algorithm updates the velocity and position of each particle as follows [17]:

$$s_t^{[m]} = s_{t-1}^{[m]} + v_t^{[m]} \quad (45)$$

$$v_t^{[m]} = |\mathrm{rand}n|\left(P_{\mathrm{pbest}} - s_{t-1}^{[m]}\right) + |\mathrm{rand}n|\left(P_{\mathrm{gbest}} - s_{t-1}^{[m]}\right). \quad (46)$$

PSO moves all particles toward the particle with the best fitness. Because PSO is an iteration algorithm in search space, it will cost much time. However, in our algorithm, PSO does not need to iterate so many times. This is because the searching space is a small area around the position at time step $t-1$ and initialized particles are also around the true position. The experiments demonstrate that five times is appropriate. At the end of iterations, P_{gbest} and P_{pbest} are calculated, and the particles are distributed according to the values of P_{gbest} and P_{pbest}. With this set of particles, the sampling process will be conducted on the basis of the proposal distribution. Fig. 1 shows the loop of sampling in the proposed method.

B. Improved Resampling of SRUFastSLAM

In general, a property of particle filters and, consequently, FastSLAM algorithms is that the variance of sample weights increases over time.

To solve this problem, resampling is performed. Although the resampling step reduces the effects of the degeneracy

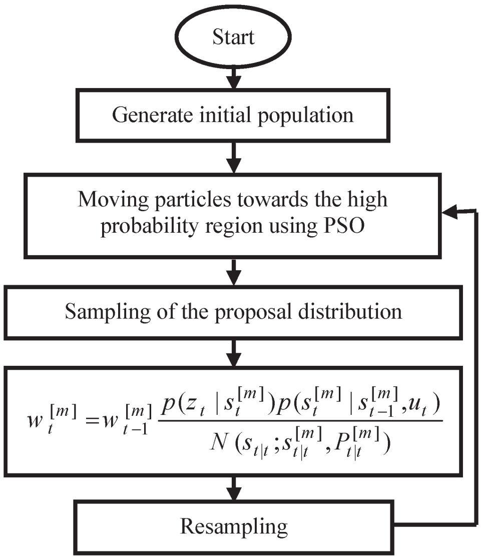

Start
Generate initial population
Moving particles towards the high probability region using PSO
Sampling of the proposal distribution
$w_t^{[m]} = w_{t-1}^{[m]} \frac{p(z_t \mid s_t^{[m]}) p(s_t^{[m]} \mid s_{t-1}^{[m]}, u_t)}{N(s_{t|t}; s_{t|t}^{[m]}, P_{t|t}^{[m]})}$
Resampling

Fig. 1. Loop of sampling in the proposed method.

problem, it introduces other practical problems in the FastSLAM algorithm. It can delete good samples from the sample set, and in the worst case, the filter diverges. In addition, whenever resampling is performed in FastSLAM, an entire pose history and map hypothesis is lost forever. This leads to a loss of diversity among the particles representing past poses and consequently erodes the statistics of the landmark estimates conditioned on these past poses. As time goes by, the number of distinct particles decreases, and consequently, the covariance of samples will decrease. This will eventually lead to filter inconsistency.

To overcome this problem, it is important to determine when and how the resampling must be performed. Grisetti *et al.* introduced the effective number of particles N_{eff} to estimate how well the current particle set represents the true posterior [16]. The resampling process is operated whenever N_{eff} is below a predefined threshold N_{if}. In this paper, we use this criterion for resampling. In the field of particle filters, many resampling algorithms have been researched for performing resampling. The most representative resampling algorithms are multinomial resampling [18], stratified resampling [19], [20], systematic resampling [18], [20], and the residual resampling [18], [19], [21]. In FastSLAM algorithms, the usual resampling technique is the RSR technique which uses weight strata to decide how many copies of each particle should be made. This technique is used in many research works [10], [11]. Although the RSR technique has a fine performance, it causes the loss of particle diversity. In order to restrain the loss of diversity, the adaptive partial rank-based resampling (APRR) algorithm is introduced.

1) APRR: The idea of APRR is to perform resampling only on particles with large weights and replace them with particles with negligible weights. Particles with moderate weights are not resampled. The APRR methods consist of two steps: 1) The particles are classified as moderate, negligible, or dominating, and 2) the number of times that each particle is replicated is determined.

TABLE I
RANK EACH PARTICLE

Particle	weight	rank
1	0.05	3
2	0.5	1
3	0.3	2

In the first step of APRR, the weight of each particle is compared with high and low thresholds ω_h and ω_l, respectively. Particles with weights between these two thresholds are considered moderate and are not resampled. Let the number of particles with weights greater than ω_h and less than ω_l be denoted by M_h and M_l, respectively. A sum of the weights of the particles that are resampled is computed using $S_{hl} = \sum_{i=1}^{M_h+M_l} w_t^{[i]}$, where i is chosen so that the condition $w_t^{[i]} > w_h$ or $w_t^{[i]} < w_l$ is satisfied.

In the second step of APRR, the ranks of particles whose weights satisfy $w_t^{[i]} > w_h$ or $w_t^{[i]} < w_l$ are determined by their weights. For example, suppose that, in a population of size 3, the weight values of particles are {0.05, 0.5, 0.3}. The ranks of particles are determined as in Table I.

In this case, the particle with the highest weight value has rank 1, and the ranks of other particles are also determined according to their weights. Then, APRR gives each particle a selection probability. In a population size of n_p, if the rank of the least fit individual is defined as zero and that of the best fit individual is defined as $n_p - 1$, then the selection probability for individual i is defined as follows:

$$p(i) = \frac{\alpha_{\text{rank}} + [\text{rank}(i)/(n_p - 1)]\,(\beta_{\text{rank}} - \alpha_{\text{rank}})}{n_p} \quad (47)$$

where α_{rank} is the number of offsprings allocated to the worst individual and β_{rank} is the expected number of offsprings to be allocated to the best individual during each generation. It is shown that, when the number of particles is fixed, $\alpha_{\text{rank}} = 2 - \beta_{\text{rank}}$ and $1 \le \beta_{\text{rank}} \le 2$. With this selection probability, one of the stratified resampling algorithms is conducted only on particles whose weights satisfy the threshold constraints. Systematic resampling is used since it is the fastest resampling algorithm for computer simulations.

As a result, in comparison with RSR, the proposed resampling method is different in structure and characteristics such as speed and diversity that is very important in computational complexity and consistency. In RSR, resampling is conducted on all particles, while APRR resampling is conducted on a part of the particles. In RSR, resampling is based on weights of particles, while in APRR, resampling is based on the selection probability. Moreover, the APRR algorithm can control the thresholds for keeping a degree of particle diversity or reducing the degeneracy. These characteristics of APRR will cause the resampling to be conducted faster as it is conducted on a smaller number of particles.

C. Computational Complexity

The computational complexity of the proposed algorithm is related to the sampling strategy, feature estimation, calculation

TABLE II
COMPARISON OF COMPLEXITY COMPUTATION

Operation	Complexity	
	Proposed Method	UFastSLAM and FastSLAM
Moving particles	O($\bar{M}$)	---
Sampling	O($\bar{M}$)	O(M)
Feature estimation	O($\bar{M}$)	O(M)
Importance weight	O($\bar{M}$)	O(M)
Resampling	O((M_1+M_h)logN)	O(MlogN)

TABLE III
COMPUTATIONAL COST OF ALGORITHMS

Number of Particles	RMSE		Processing time	
	Proposed Method	UFastSLAM	Proposed Method	UFastSLAM
30	0.04	0.109	63.2	65.4
25	0.042	0.1302	51.5	53.2
20	0.0473	0.1599	41.7	42.6
15	0.049	0.19	30.1	31
10	0.0503	0.2409	19.6	21.3

of the importance weight, and resampling. As the computational complexity of SRUKF is equivalent to UKF and EKF [15], the number of particles $\bar{M}$, the number of features N, and the number of iterations in PSO k determine the complexity of the proposed algorithm. In the proposed algorithm, the complexity of moving particles toward the high probability region using PSO is O$(K\bar{M})$ that can be approximated to O$\bar{M}$ as PSO does not need to iterate so many times, as mentioned previously. In addition, computing the proposal distribution, calculating the importance weights, and updating the landmarks have a complexity of $O(\bar{M})$. Also, the complexity of resampling is O$((M_l + M_h)N)$ or O$((M_l + M_h)\log N)$ when the landmarks are represented by a binary tree. Table II depicts the complexity of the individual operations in algorithms where M is the number of particles in FastSLAM and UFastSLAM. As it can be seen from Table II, the proposed algorithm has an additional operation (i.e., moving particles toward the high probability region). However, in the proposed algorithm, resampling is conducted on a part of the particles (i.e., $M_l + M_h < M$), and as a result, the complexity of resampling is less than that of UFastSLAM and FastSLAM. Moreover, the proposed method requires fewer particles to obtain the same estimation accuracy compared to UFastSLAM and FastSLAM (i.e., $\bar{M} < M$). For example, as will be shown in Table III in Section V, the estimation accuracy of the proposed method with 10 particles (i.e., $\bar{M} = 10$) is better than that of UFastSLAM with 30 particles (i.e., $M = 30$). This means that the complexity of other operators is less than that of UFastSLAM and FastSLAM. Therefore, considering the complexity of the additional operator, the complexity of the proposed algorithm is almost the same as those of UFastSLAM and FastSLAM.

V. RESULTS

The proposed algorithm is evaluated on simulated data and real world data sets. The simulated data allow comparison with ground truth, while the real world data prove the applicability of the proposed approach to practical problems.

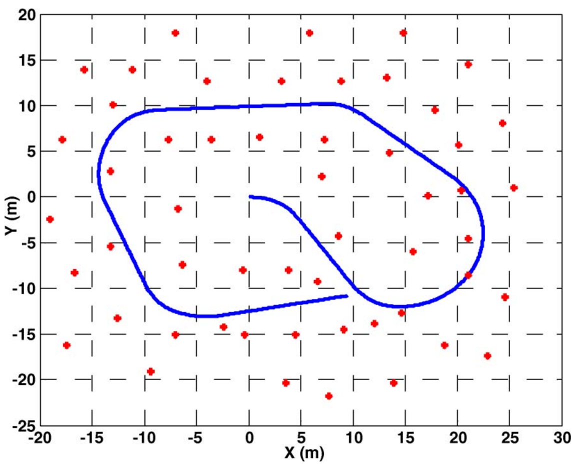


Fig. 2. Experiment environment.

A. Simulation Results

Simulation has been carried out to evaluate the performance of the proposed approach in comparison with UFastSLAM for the benchmark environment, available in [22]. Fig. 2 shows the robot trajectory and landmark location. The star points $(*)$ depict the location of the landmarks that are known and stationary in the environment. The robot moves at a speed of 3 m/s and with a maximum steering angle of 30°. The robot also has a 4-m wheel base and is equipped with a range-bearing sensor with a maximum range of 20 m and a 180° frontal field of view. The control noise is (0.3 m/s, 3°), and the measurement noise is (0.1, 1°). The control frequency is 40 Hz, and observation scans are obtained at 5 Hz.

1) Performance Comparison of Proposed Method and UFastSLAM: At first, we compare the performance of the proposed algorithm with that of UFastSLAM while the measurement noise is $(\sigma_r = 0.1, \sigma_\theta = 1)$ and the control noise is $(\sigma_r = 0.3, \sigma_\theta = 3)$. In this experiment, the number of particles is 30. Also, the results are obtained over 30 Monte Carlo runs.

Fig. 3 shows the estimated robot path and estimated landmark with the true robot path and true landmarks. The root-mean-square error (RMSE) of the position robot over time is shown in Fig. 4, and its mean and variance are shown in Fig. 5. Each bar in Fig. 5 represents the mean and variance of RMSE for the robot position. It is observed that the performance of the proposed method is better than that of UFastSLAM. This is because sampling and resampling are improved in the proposed method. In the proposed method, PSO merges into the proposal distribution to improve the distribution samples and speed up the convergence of the particle set. Therefore, the effect of each particle is enhanced, and the diversity is improved. On the other hand, SRUKF is able to estimate the mean and the covariance to a higher order of accuracy and consistency than UKF. Finally, the new resampling method increases the diversity of particles and thus increases the accuracy and consistency of estimation.

Now, the performances of the two algorithms are compared while the level of measurement noise is changed. The mean

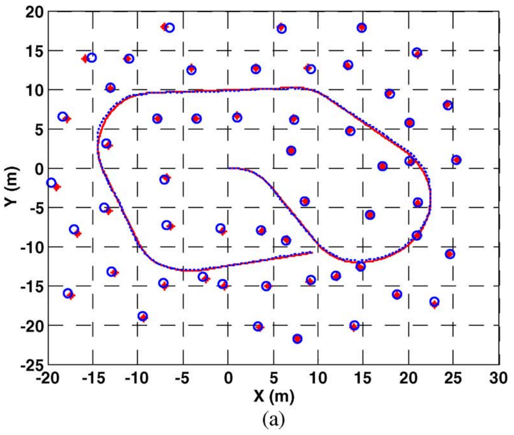
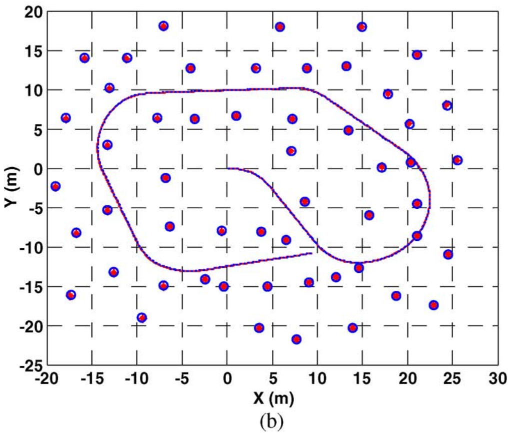



Fig. 3. Estimated robot path and estimated landmark with true robot path and true landmarks. The "..." is the estimated path, and the "o" denotes the estimated landmark positions: (a) UFastSLAM and (b) proposed method.

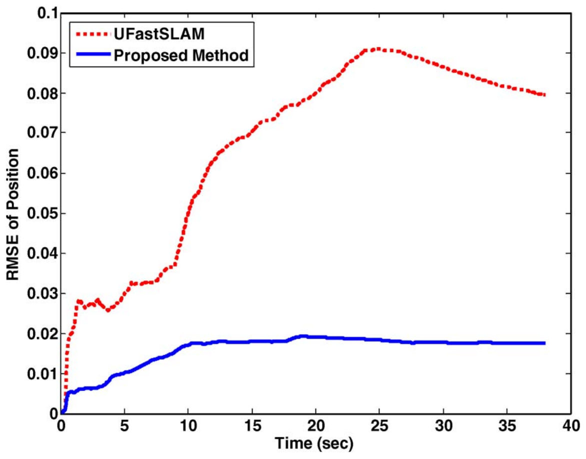


Fig. 4. RMSE with respect to time.

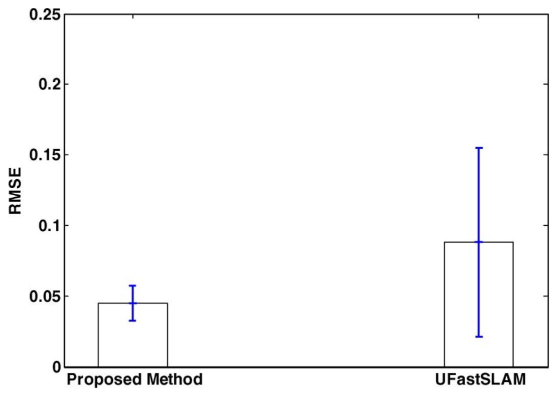


Fig. 5. RMSE for the robot position.

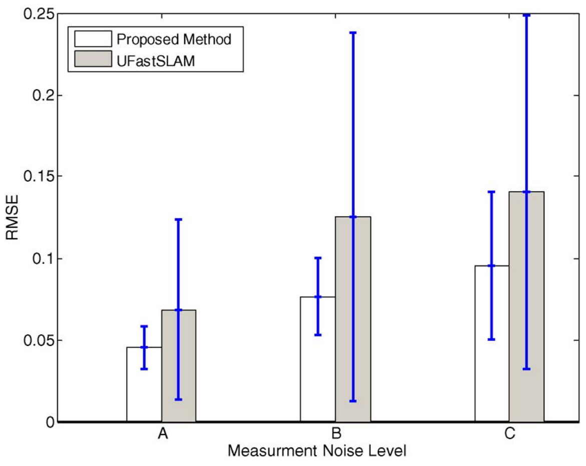


Fig. 6. Performances of algorithms with varying levels of measurement noise. Measurement noises of each experiment are as follows: (A) $(\sigma_r = 0.1, \sigma_\theta)$, (B) $\sigma_r = 0.5, \sigma_\theta = 3$, and (C) $(\sigma_r = 0.9, \sigma_\theta = 4)$.

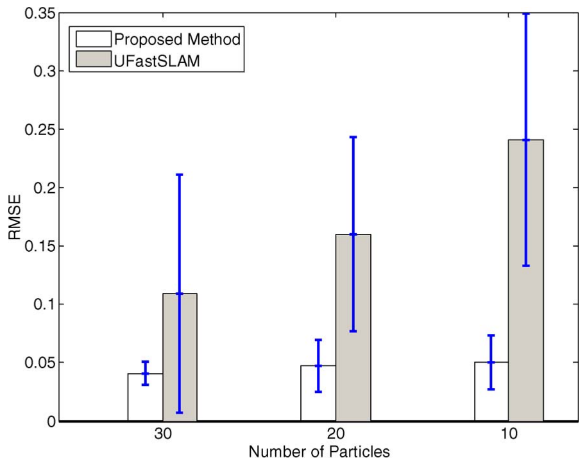


Fig. 7. Performance of algorithms with different numbers of particles.

and variance of RMSE are calculated over ten independent runs for each algorithm. The results of this experiment are shown in Fig. 6. As the measurement noise is increased, the RMSE of the proposed method and the UFastSLAM algorithm increases. However, the mean and variance of RMSE for the proposed method are smaller than those of UFastSLAM. Thus, the proposed method is more robust than UFastSLAM while varying the level of measurement.

Next, the proposed algorithm is compared with UFastSLAM with various numbers of particles. Fig. 7 shows the results for this case. It is observed that the performance of the proposed method does not heavily depend on the number of particles, while the performance of UFastSLAM depends on the number of particles. This is because PSO places the particles in the proprietary region before sampling in the proposed method. In addition, the new resampling algorithm maintains the diversity of particles and makes the resampled particles asymptotically approximate the samples from the posterior pdf of the true state.

That is, the proposed algorithm can avoid the sample impoverishment; thus, there is no need to sample a large number of particles for compensation. The advantage of the new algorithm is that it can decrease the computational cost to a certain extent since resampling is conducted only on a part of the particles.

2) Computational Cost: The computational cost of the proposed algorithm is analyzed using Matlab simulations on an Intel Core2Duo@3GHz PC. Although PSO increases the computational time of sampling, the required computational time of the proposed method is approximately close to that of UFastSLAM, as shown in Table III. This is because the proposed approach uses SRUKF to generate the importance density, initialization, and estimation of features. SRUKF is faster than UKF, while their computational complexities are the same. In addition, resampling is conducted on a smaller number of particles. The main advantage of the proposed method from a computational cost viewpoint is that, to obtain the same estimation accuracy as that of UFastSLAM, a lower number of particles is needed. Table III shows that, when one-third of the particles of UFastSLAM (10 particles) is used, the estimated error is smaller than that of UFastSLAM. Therefore, the proposed algorithm gains more accuracy with a lower computational cost.

3) Consistency of Proposed Method: To verify the consistency, the average normalized estimation error squared (NEES) is used. For an available ground truth s_t and the estimated mean and covariance $\{\hat{s}_t \text{ and } \hat{P}_t\}$, we can use NEES to characterize the filter performance [23]

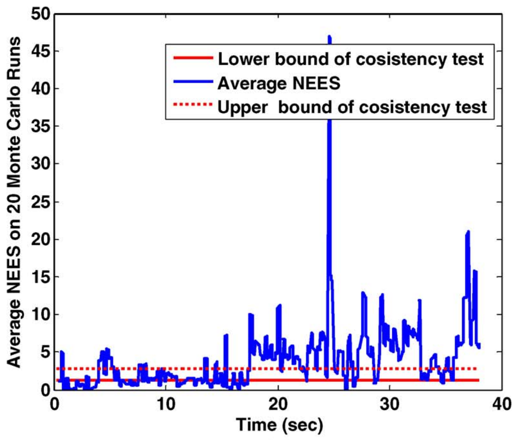
$$\varepsilon_t = (s_t - \hat{s}_t)^T P_t^{-1} (s_t - \hat{s}_t). \tag{48}$$

Consistency is evaluated by performing multiple Monte Carlo runs and computing the average NEES. Given N_R runs, the average NEES is computed as

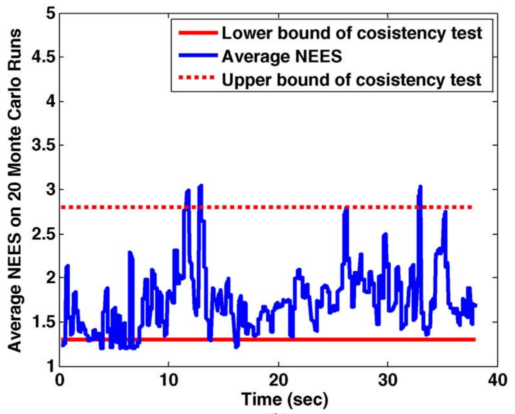
$$\bar{\varepsilon}_t = \frac{1}{N_R} \sum_{i=1}^{N_R} \varepsilon_{it}. \tag{49}$$

Given the hypothesis of a consistent linear-Gaussian filter, $N_R\bar{\varepsilon}_t$ has a χ^2 density with $N_R \dim(s_t)$ degrees of freedom. Thus, for the 2-D vehicle position, with 20 Monte Carlo simulations, the two-sided 95% probability concentration region for $\bar{\varepsilon}$ is bounded by interval [1.3, 2.79]. Fig. 8 shows that the consistency of the proposed method is more than that of UFastSLAM. This is because the diversity of particles in the proposed method is more than that of UFastSLAM. An estimate of the rate of loss of particle diversity is obtained by recording the number of distinct particles. Fig. 9 shows the number of distinct particles, which are counted after every resampling for the proposed method and UFastSLAM. The result shows that the number of distinct particles in the proposed method is more than that of UFastSLAM. Hence, the consistency of the proposed method is more than that of UFastSLAM.

One advantage of the proposed method from a consistency viewpoint is that its consistency is more than that of UFastSLAM for a large loop. As the size of the loop increases, the errors of both algorithms are almost equal, as shown in Fig. 10. However, the proposed algorithm consistently outperforms UFastSLAM, as shown in Fig. 11. This is because of better proposal distribution and resampling. The better proposal distribution and the resampling algorithm cause more diversity and, consequently, consistency in the proposed algorithm.



(a)



(b)

Fig. 8. Consistency: (a) UFastSLAM and (b) proposed method.

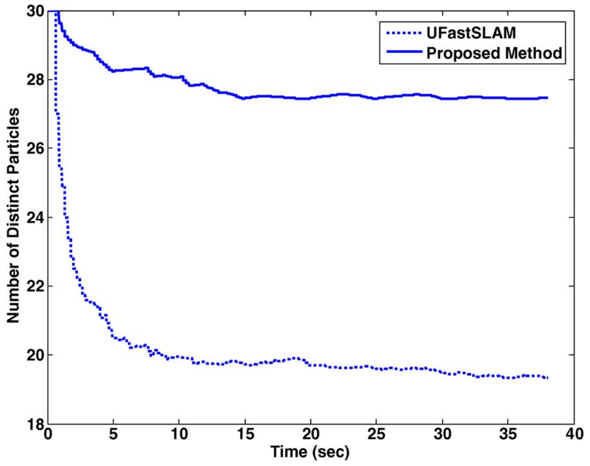


Fig. 9. Number of distinct particles.

B. Experimental Results

The proposed algorithm is compared to UFastSLAM using the Car Park data set and Victoria data set, two popular data sets in the SLAM community. These two data sets are available in [24]. The experimental platform was a four-wheeled vehicle

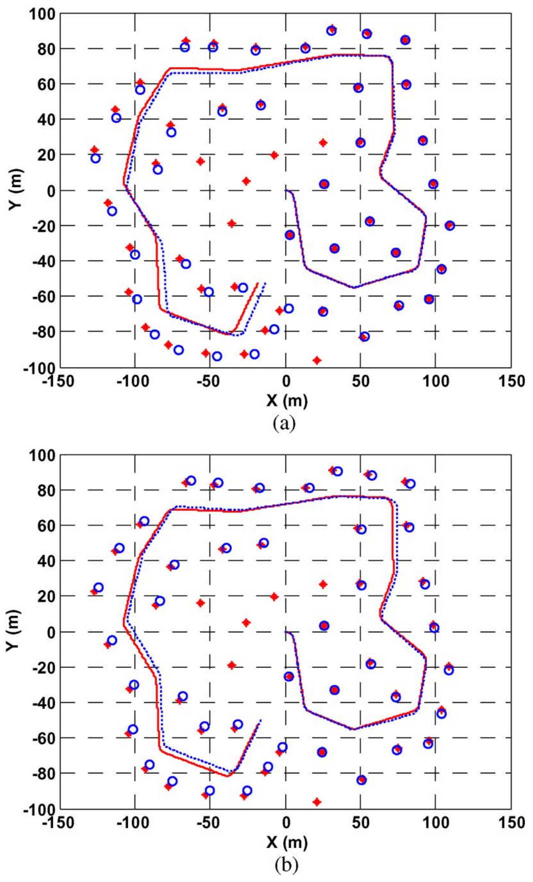


Fig. 10. Estimated robot path and estimated landmark with true robot path and true landmark. The "..." is the estimated path, and the "o" denotes the estimated landmark positions: (a) UFastSLAM and (b) proposed method.

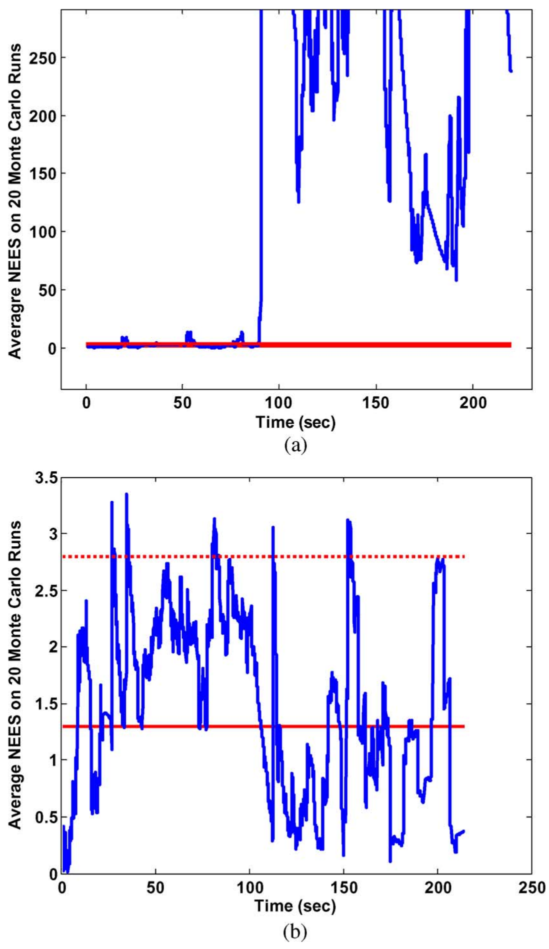


Fig. 11. Consistency: (a) UFastSLAM and (b) proposed method.

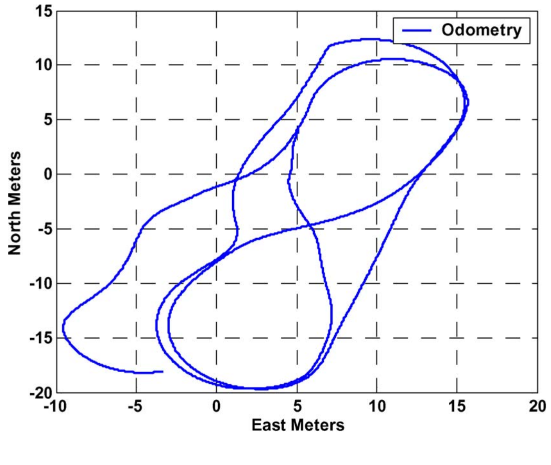


Fig. 12. Odometry of the vehicle.

equipped with wheel encoders, GPS, and a laser sensor. The vehicle had a 2.83-m wheel base and was equipped with the SICK laser range finder with a 180° frontal field of view.

In the Car Park test, the vehicle was driven around the park. The steering angle and the velocity were measured using encoders, but uneven terrain induced additional nonsystematic errors because of wheel slippage and attitude errors. Consequently, the odometry information from the encoder is poor as shown in Fig. 12. The nature of the terrain created additional errors in the vehicle prediction since wheel slip and attitude errors are not taken into account in the prediction models of our current implementation. The artificial landmarks were used that consisted of 60-mm steel poles covered with reflective tape. With this approach, the feature extraction becomes trivial, and the landmark observation model is more accurate. Since the true position of the landmarks was also obtained with GPS, a true navigation map was available for comparison purposes. Also, a kinematic GPS system is used to provide ground truth for the robot position.

The performance of the proposed algorithm is compared with that of the classical UFastSLAM in a situation that the correspondences between the observation and the landmarks were assumed to be unknown and 20 particles were used for both algorithms. Each algorithm has been executed 20 times to confirm the variance of the estimated error. For the unknown data association, the individual compatibility nearest neighbor test with a 2σ acceptance region is used. Fig. 13 shows the trajectory and landmark estimates produced by the proposed approach and UFastSLAM, while Fig. 14 shows the RMSE errors of the robot position for the two algorithms. The results show that the performance of the proposed algorithm is better than that of UFastSLAM.

The second experimental run was performed in the Victoria data set that is a larger area with mild uneven terrain and different types of surfaces. The vehicle was driven around in the park for about 30 min and moved over 4 km, with a

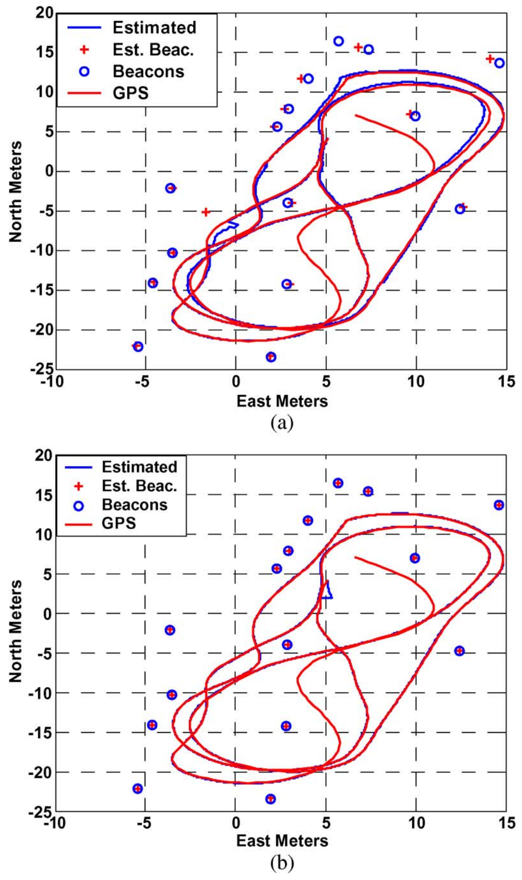


Fig. 13. (a) UFastSLAM. (b) Proposed method. The "..." is the path estimated, the "+" denotes the estimated beacon positions, the "__" is the GPS path reference, and the "o" denotes the beacon positions given by the GPS.

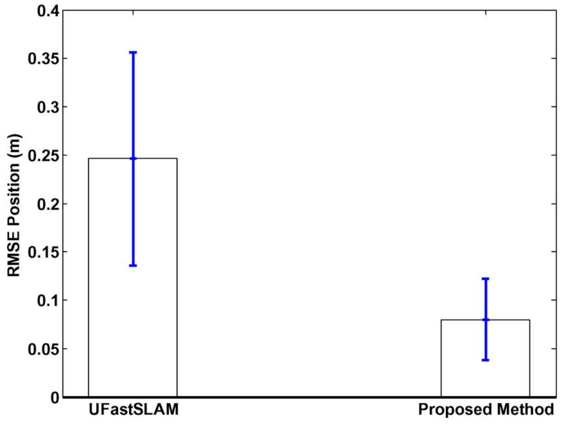


Fig. 14. RMSE for the robot position.

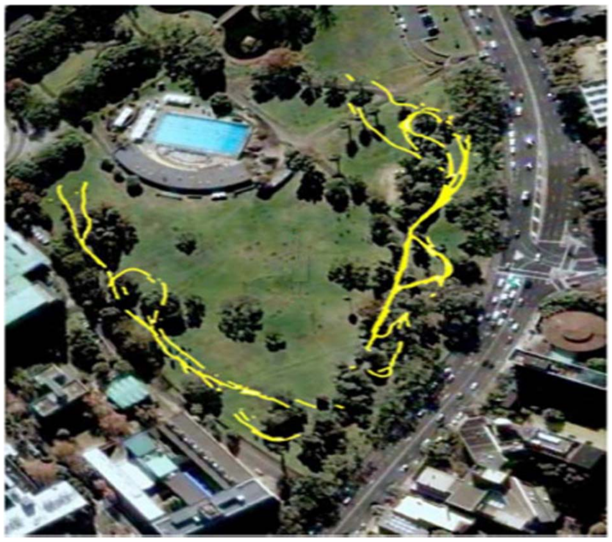


Fig. 15. Victoria Park with the GPS path.

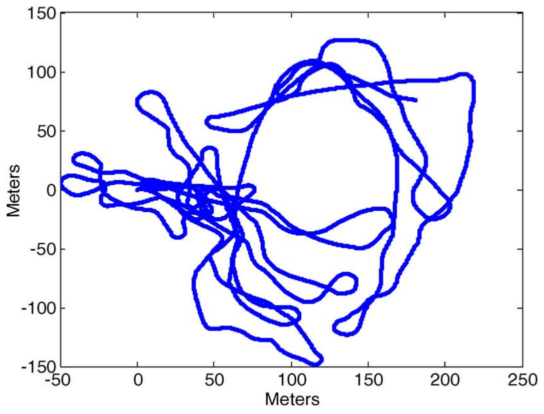


Fig. 16. Trajectory based on odometry only.

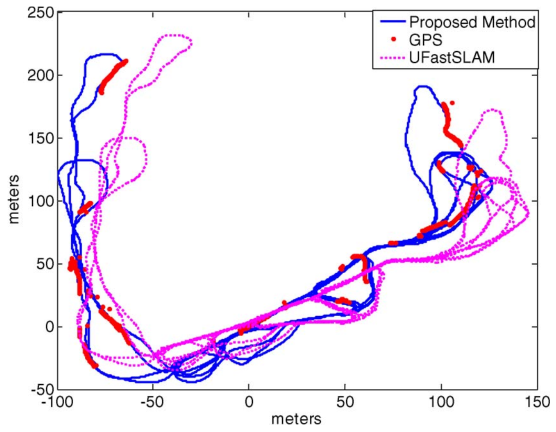


Fig. 17. Comparison of UFastSLAM and the proposed method.

sensor to measure the velocity and the steering angle. Fig. 15 shows the Victoria Park with the GPS path. GPS data were collected to provide ground-truth data. Fig. 16 shows the trajectory based on odometry only. Since the robot is driving over uneven terrain, the odometry information from the encoder is poor. Although the vehicle was equipped with GPS, due to occlusion by foliage and buildings, ground-truth data were not available over the whole experiment. Nevertheless, the ground true position of the vehicle from the GPS was good enough to validate the estimated vehicle state. Each algorithm has been executed independently several times to verify the variance and the estimated error, as well as the comparison results of accuracy for each approach. Fig. 17 presents the trajectories for the proposed approach and UFastSLAM. The results show that the performance of the proposed algorithm is better than that of UFastSLAM. This is because the estimated trajectory of the proposed method coincides very well with GPS paths. As the proposal distribution and resampling step are optimized in the proposed approach, the uncertainties are propagated well, and the accuracy of the state estimation has been improved over that of UFastSLAM.

VI. Conclusion

UFastSLAM has been recently proposed for solving the SLAM problem. However, this approach can only be achieved consistently for longer time periods. In addition, one of the most costly operations in UFastSLAM is the calculation of the matrix square root of the covariance of the state variable at each time in order to form the square set. In this paper, we proposed SRUFastSLAM with improved proposal distribution and resampling. Simulations and experimental results indicate that the proposed algorithm can improve the accuracy of estimation evidently and maintain the diversity of particles. The main advantage of the proposed method is that, to obtain the same estimation accuracy as that of UFastSLAM, a lower number of particles is needed. In addition, the consistency of this approach is higher than that of UFastSLAM.

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[2] S. A. Holmes, G. Klein, and D. W. Murray, "An O (N^2) square root unscented Kalman filter for visual simultaneous localization and mapping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 7, pp. 1251–1263, Jul. 2009.

[3] S. Y. Hwang and J. B. Song, "Monocular vision-based SLAM in indoor environment using corner, lamp, and door features from upward-looking camera," *IEEE Trans. Ind. Electron.*, vol. 58, no. 10, pp. 4804–4812, Oct. 2011.

[4] X. Yan, C. Zhao, and J. Xiao, "A novel FastSLAM algorithm based on iterated unscented Kalman filter," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2011, pp. 1906–1911.

[5] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *J. Mach. Learn. Res.*, vol. 4, no. 3, pp. 380–407, 2004.

[6] Z. Q. Wei, J. Cao, B. Yin, and B. Liu, "Improved FastSLAM based on the particle fission for mobile robots," in *Proc. IEEE Int. Conf. Control Autom.*, 2010, pp. 1379–1384.

[7] I. Kim, N. Kwak, H. Lee, and B. Lee, "Improved particle fusing geometric relation between particles in FastSLAM," *J. Robot.*, vol. 27, no. 6, pp. 853–859, Oct. 2009.

[8] C. Qiu, X. Zhu, and X. Zhao, "Vision-based unscented FastSLAM for mobile robot," in *Proc. IEEE World Congr. Intell. Control Autom.*, 2012, pp. 3758–3763.

[9] Z. Kurt-Yavuz and S. Yavuz, "A comparison of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM algorithms," in *Proc. IEEE Int. Conf. Intell. Eng. Syst.*, 2012, pp. 37–43.

[10] C. Kim, H. Kim, and W. K. Chung, "Exactly Rao-Blackwellized unscented particle filters for SLAM," in *IEEE Int. Conf. Robot. Autom.*, 2011, pp. 3589–3594.

[11] C. Kim, R. Sakthivel, and W. K. Chung, "Unscented FastSLAM: A robust and efficient solution to the SLAM problem," *IEEE Trans. Robot.*, vol. 24, no. 4, pp. 808–820, Aug. 2008.

[12] S. Julier, J. Uhlmann, and F. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 477–482, Mar. 2000.

[13] V. Smidl and Z. Peroutka, "Advantages of square-root extended Kalman filter for sensorless control of ac drives," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4189–4196, Nov. 2012.

[14] N. He, D. Xu, and L. Huang, "The application of particle swarm optimization to passive and hybrid active power filter design," *IEEE Trans. Ind. Electron.*, vol. 56, no. 8, pp. 2841–2851, Aug. 2009.

[15] R. van der Merwe and E. Wan, "The square-root unscented Kalman filter for state and parameter estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2001, vol. 6, pp. 3461–3464.

[16] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[17] R. Krohling, "A Gaussian swarm: A novel particle swarm optimization algorithm," in *Proc. IEEE Conf. CIS*, Singapore, 2004, pp. 372–376.

[18] C. Fen, M. Wang, and Q. B. Ji, "Analysis and comparison of resampling algorithms in particle filter," *J. Syst. Simul.*, vol. 21, no. 4, pp. 1101–1105, 2009.

[19] J. X. Yu, Z. X. Cai, and Z. H. Duan, "Survey on some key technologies of mobile robot localization based on particle filter," *J. Appl. Res. Comput.*, vol. 24, no. 11, pp. 9–14, 2007.

[20] Z. C. Du, Research on particle and its application in MIMO wireless communication, Chengdu, China 2008.

[21] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2442–2450, Jul. 2005.

[22] [Online]. Available: http://www-personal.acfr.usyd.edu.au/tbailey

[23] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation*. Hoboken, NJ, USA: Wiley, 2001.

[24] E. Nebot, Victoria Park Data Set. [Online]. Available: http://www-personal.acfr.usyd.edu.au/nebot/dataset.htm

Ramazan Havangi received the M.Sc. degree in control engineering from the K. N. Toosi University of Technology, Tehran, Iran, in 2003, where he is currently working toward the Ph.D. degree in control engineering.

His current research interests include inertial navigation, integrated navigation, estimation and filtering, evolutionary filtering, simultaneous localization and mapping, fuzzy, neural network, and soft computing.

Hamid D. Taghirad (S'96–M'97–SM'12) received the B.Sc. degree in mechanical engineering from the Sharif University of Technology, Tehran, Iran, in 1989 and the M.Sc. degree in mechanical engineering and the Ph.D. degree in electrical engineering from McGill University, Montreal, QC, Canada, in 1993 and 1997, respectively.

He is currently a Professor with the Faculty of Electrical and Computer Engineering of the Department of Systems and Control and the Director of the Advanced Robotics and Automated System at the K. N. Toosi University of Technology, Tehran. He is the Editor in Chief of *Mechatronics Magazine*. His research interests include robust and nonlinear control applied to robotic systems.

Dr. Taghirad is a member of the board of the Industrial Control Center of Excellence at the K. N. Toosi University of Technology.

Mohammad Ali Nekoui received the M.Sc. degree from the University of Tehran, Tehran, Iran, in 1976, and the Ph.D. degree from the University of Leeds, Leeds, U.K., in 1997, both in electrical engineering.

He is currently an Associate Professor of control systems with the Department of Electrical and Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran. His research interest includes linear and nonlinear optimization, linear systems, optimal control, and different aspects of mathematics in control.

Mohammad Teshnehlab received the M.S. degree in electrical engineering from Oita University, Japan, in 1991, and the Ph.D. degree in computational intelligence from Saga University, Japan, in 1994.

He is currently a Professor of control systems with the Department of Electrical and Computer Engineering, K. N. Toosi University of Technology, Tehran, Iran. His main research interests are neural networks, fuzzy systems, evolutionary algorithms, swarm intelligence, fuzzy-neural networks, pattern recognition, metaheuristic algorithms, and intelligent identification, prediction, and control.