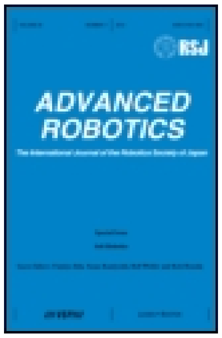

Advanced Robotics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tadr20

An intelligent UFastSLAM with MCMC move step

Ramazan Havangi [a] , Mohammad Ali Nekoui [a] , Hamid D. Taghirad [a] & Mohammad Teshnehlab [a]

[a] Faculty of Electrical and Computer Engineering, Department of Systems and Control, K.N. Toosi University of Technology, Tehran, Iran

Version of record first published: 05 Feb 2013.

To cite this article: Ramazan Havangi , Mohammad Ali Nekoui , Hamid D. Taghirad & Mohammad Teshnehlab (2013): An intelligent UFastSLAM with MCMC move step, Advanced Robotics, DOI:10.1080/01691864.2013.763721

To link to this article: http://dx.doi.org/10.1080/01691864.2013.763721

Advanced Robotics, 2013
http://dx.doi.org/10.1080/01691864.2013.763721

FULL PAPER

An intelligent UFastSLAM with MCMC move step

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

Faculty of Electrical and Computer Engineering, Department of Systems and Control, K.N. Toosi University of Technology, Tehran, Iran

(*Received 29 February 2012; accepted 24 July 2012*)

FastSLAM is a framework for simultaneous localization and mapping (SLAM). However, FastSLAM algorithm has two serious drawbacks, namely the linear approximation of nonlinear functions and the derivation of the Jacobian matrices. For solving these problems, UFastSLAM has been recently proposed. However, UFastSLAM is inconsistent over time due to the loss of particle diversity that is caused mainly by the particle depletion in the resampling step and incorrect a priori knowledge of process and measurement noises. To improve consistency, intelligent UFastSLAM with Markov chain Monte Carlo (MCMC) move step is proposed. In the proposed method, the adaptive neuro-fuzzy inference system supervises the performance of UFastSLAM. Furthermore, the particle impoverishment caused by resampling is restrained after the resample step with MCMC move step. Simulations and experiments are presented to evaluate the performance of algorithm in comparison with UFastSLAM. The results show the effectiveness of the proposed method.

Keywords: UFastSLAM; Markov chain Monte Carlo; adaptive neuro-fuzzy inference system (ANFIS)

1. Introduction

The simultaneous localization and mapping (SLAM) is a fundamental problem of mobile robots to perform autonomous tasks such as exploration in an unknown environment. It represents an important role in the autonomy of a mobile robot. The two key computational solutions to the SLAM problem are EKF (extended Kalman filter)-SLAM and FastSLAM.[1] The EKF-SLAM approach is the most popular one to solve the SLAM problem. However, EKF-SLAM suffers from two major problems: the computational complexity and data association.[1]

Recently, the FastSLAM algorithm has been introduced as an approach to solve the SLAM problem.[1–3] The two versions of FastSLAM can be found in literatures, namely FastSLAM1.0 and FastSLAM2.0.[1,3] In the FastSLAM algorithm, particle filter (PF) is used to estimate the robot pose, and EKF is used to estimate the location of the landmarks.[1] The key feature of FastSLAM is that the data association decisions can be determined on a per-particle basis, and hence different particles can be associated with different landmarks. Each particle in FastSLAM may even have a different number of landmarks in its respective map.[1–3] This characteristic gives FastSLAM the possibility of dealing with the multi-hypothesis association problem. The ability to pursue multiple data associations makes FastSLAM significantly more robust than EKF-SLAM.[1] The other advantage of FastSLAM over EKF-SLAM is that PFs can cope with nonlinear and non-Gaussian robot motion models, whereas EKF approaches approximate such models via linear functions.

There have been many researches on FastSLAM.[4–9] However, FastSLAM has some drawbacks which are the derivation of the Jacobian matrices and the linear approximations of the nonlinear functions.[9–13] To overcome these problems, a number of authors have proposed UFastSLAM.[9–13] UFastSLAM overcomes the drawbacks caused by linearizations in the FastSLAM framework. In UFastSLAM, the linearization process with Jacobian calculations is removed by applying the unscented transformation.[14,15] This approach can achieve more consistency for longer time periods with respect to FastSLAM.[8,9] However, the resampling process decreases the diversity of particles by throwing away some particles and duplicating others multiple times. The resampling step causes the particles to share a common history, and new observations cannot affect the locations of observed features.[4–6] Thus, keeping the diversity of particles is important for reliable loop closing and consistent map building in UFastSLAM. When particles lose their diversity, they tend to underestimate their own uncertainty, and this will cause the inconsistency of UFastSLAM. Therefore, we require a procedure to introduce sample variety after the resampling step without affecting the validity of the posterior estimation. Moreover, unscented Kalman filter (UKF) and consequently

*Corresponding author. Email: rhavangi@gmail.com, havangi@kntu.ac.ir

UFastSLAM can only achieve suitable consistency under the assumption that some information has to be known as a priori of the process and measurement noise covariances. In real-life application, these matrices are unknown and on the other hand, it is well known that the poor estimations of noise statistics may seriously degrade the UKF performance.[16–20]

To overcome the drawbacks of UFastSLAM, this paper proposes a new probabilistic framework called intelligent UFastSLAM with Markov chain Monte Carlo (MCMC) move step. In this approach, adaptive neuro-fuzzy inference system (ANFIS) supervises the performance of UFastSLAM and MCMC move step is used to increase the diversity of the particles after the resampling step.

The rest of the paper is as follows: Section 2 introduces the SLAM problem and reviews UFastSLAM. Section 3 presents the proposed algorithm. The effectiveness of the proposed algorithm is demonstrated using simulation and experimental results in Section 4.

2. Background: UFastSLAM

To describe SLAM, let us denote the robot pose at time t by s_t and the map by Θ. The map 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 SLAM problem can be formulized in a Bayesian probabilistic framework as [1]:

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

where $s^t = \{s_1, \ldots, s_t\}$ is the robot path, $z^t = \{z_1, \ldots, z_t\}$ is the observation, $u^t = \{u_1, \ldots, u_t\}$ is the control input, and $n^t = \{n_1, \ldots, n_t\}$ is the data association. FastSLAM is an efficient algorithm for the SLAM problem that is based on a straightforward factorization as [1–3]:

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

The FastSLAM algorithm implements the robot path posterior $p(s^t|z^t, u^t, n^t)$ using a PF, and N the landmark posteriors $p(\theta_n|s^t, z^t, u^t, n^t)$ are realized by a parametric filter (i.e. EKF or UKF). Each particle in FastSLAM-is composed of a robot path and a set of feature locations with their covariance as:

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

where $[m]$ Z is the index of the particles, $s^{t,[m]}$ is the mth particle's path estimate, $\mu_{N,t}^{[m]}$ and $\Sigma_{N,t}^{[m]}$ are the mean and the covariance of the Gaussian distribution representing the Nth feature location, attached to the mth particle. In UFastSLAM, a new probabilistic framework is presented to overcome the drawbacks caused by linearization in FastSLAM.[9] UFastSLAM computes the proposal distribution by measurement updates of the UKF in the sampling step and updates each feature state by UKF without calculating the Jacobian matrix of an observation model.[9] The algorithm of UFastSLAM consists of the robot pose estimation, feature estimation, feature initialization, and calculation of the importance weights. The details of UFastSLAM are summarized in the following subsections.

2.1. *Robot pose estimation*

In UFastSLAM, the robot pose s_t is sampled as:

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

where q is the proposal distribution. To generate the proposal distribution, an augmented state and augmented covariance matrix is formed by appending the mean and covariance of process noise vector. Assuming the mean of process noise is zero, the augmented state and covariance matrix is as follows:

$$s_{t-1|t-1}^{a[m]} = \begin{bmatrix} s_{t-1|t-1}^{[m]} \\ 0 \end{bmatrix} \; P_{t-1|t-1}^{a[m]} = \begin{bmatrix} P_{t-1|t-1}^{[m]} & 0 \\ 0 & Q_t \end{bmatrix} \tag{5}$$

where $s_{t-1|t-1}^{a[m]}$ is the augmented state, $P_{t-1|t-1}^{a[m]}$ is the augmented covariance matrix, Q_t is the control noise covariance, $s_{t-1|t-1}^{[m]}$ is the mean and $P_{t-1|t-1}^{[m]}$ is the covariance of the robot pose at time $t-1$. A symmetric set of $2n+1$ sigma points is sampled from the augmented state as:

$$\begin{aligned}
\delta_{t-1|t-1}^{a[0][m]} &= s_{t-1|t-1}^{a[m]} \\
\delta_{t-1|t-1}^{a[i][m]} &= s_{t-1|t-1}^{a[m]} + \left(\sqrt{(n+\lambda) P_{t-1|t-1}^{a[m]}} \right)_i \qquad i = 1, \ldots, n \\
\delta_{t-1|t-1}^{a[i][m]} &= s_{t-1|t-1}^{a[m]} - \left(\sqrt{(n+\lambda) P_{t-1|t-1}^{a[m]}} \right)_i \qquad i = n+1, \ldots, 2n
\end{aligned} \tag{6}$$

where the term $\left(\sqrt{(n+\lambda) P_{t-1|t-1}^{a[m]}} \right)_i$ represents ith column of the matrix $\sqrt{(n+\lambda) P_{t-1|t-1}^{a[m]}}$, n is the dimension of augmented state and λ is a scaling parameter. Each sigma point $\delta_{t-1|t-1}^{a[i][m]}$ contains the robot pose and a control noise, which can be represented as follows:

$$\delta_{t-1|t-1}^{a[i][m]} = \begin{bmatrix} \delta_{t-1|t-1}^{[i][m]} & \delta_{t-1}^{u[i][m]} \end{bmatrix}^T \tag{7}$$

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

$$\bar{s}_{t|t-1}^{[i][m]} = f\left(\delta_{t-1|t-1}^{[i][m]}, u_t + \delta_{t-1}^{u[i][m]}\right) \quad (8)$$

where f is the nonlinear motion function and $\bar{s}_{t|t-1}^{[i][m]}$ is its transformed sigma point. The transformed points are used to compute the predictive mean $s_{t|t-1}^{[m]}$ and covariance $P_{t|t-1}^{[m]}$ of the robot pose as follows:

$$s_{t|t-1}^{[m]} = \sum_{i=0}^{2n} \omega_g^{[i]} \bar{s}_{t|t-1}^{[i][m]} \quad (9)$$

$$P_{t|t-1}^{[m]} = \sum_{i=0}^{2n} \omega_c^{[i]} \left(\bar{s}_{t|t-1}^{[i][m]} - s_{t|t-1}^{[m]}\right)\left(\bar{s}_{t|t-1}^{[i][m]} - s_{t|t-1}^{[m]}\right)^T \quad (10)$$

where the weights $\omega_g^{[i]}$ and $\omega_c^{[i]}$ are as:

$$\begin{gathered} \omega_g^{[0]} = \tfrac{\lambda}{(n+\lambda)}\, \omega_c^{[0]} = \tfrac{\lambda}{(n+\lambda)} + (1 - \alpha^2 + \beta) \\ \omega_g^{[i]} = \omega_c^{[i]} = \tfrac{\lambda}{2(n+\lambda)}\ (i = 1, \ldots, 2n) \end{gathered} \quad (11)$$

The constant α is usually set to a small positive value (e.g. $10^{-4} \le \alpha \le 1$) and the parameter β is used to incorporate the knowledge of the posterior distribution. When some features were observed, the predicted measurement $\bar{z}_t^{[m]}$ and Kalman gain $K_t^{[m]}$ are calculated as:

$$\begin{aligned} \zeta_t^{[i][m]} &= h\left(\bar{s}_{t|t-1}^{[i][m]}, \mu_{k,t-1}^{[m]}\right) \\ \bar{z}_t^{[m]} &= \sum_{i=0}^{2n} w_g^{[i]} \zeta_t^{[i][m]} \end{aligned} \quad (12)$$

$$\begin{aligned} P_{\upsilon\upsilon}^{[m]} &= \sum_{i=0}^{2n} w_c^{[i]} \left(\zeta_t^{[i][m]} - \bar{z}_t^{[m]}\right)\left(\zeta_t^{[i][m]} - \bar{z}_t^{[m]}\right)^T + R_t \\ P_{\delta\upsilon}^{[m]} &= \sum_{i=0}^{2n} \omega_c^{[i]} \left(\bar{s}_{t|t-1}^{[i][m]} - s_{t|t-1}^{[m]}\right)\left(\zeta_t^{[i][m]} - \bar{z}_t^{[m]}\right)^T \\ K_t^{[m]} &= P_{\delta\upsilon}^{[m]} \left(P_{\upsilon\upsilon}^{[m]}\right)^{-1} \end{aligned} \quad (13)$$

where $h(.)$ is the observation model. The measurement is employed to update the predicted pose mean $s_{t|t-1}^{[m]}$ and the covariance $P_{t|t-1}^{[m]}$ as follows:

$$\begin{aligned} s_{t|t}^{[m]} &= s_{t|t-1}^{[m]} + K_t^{[m]} \left(z_t - \bar{z}_t^{[m]}\right) \\ P_{t|t}^{[m]} &= P_{t|t-1}^{[m]} - K_t P_{\upsilon\upsilon}^{[m]} \left(K_t^{[m]}\right)^T \end{aligned} \quad (14)$$

where z_t is the true measurement. From the Gaussian distribution generated by the estimated mean and covariance of the vehicle, the state of each particle is sampled as:

$$s_t \sim N\left(s_t; s_{t|t}^{[m]}, P_{t|t}^{[m]}\right) \quad (15)$$

when there is no observation, the robot pose is predicted without the measurement update using (9) and (10). If many features are observed at the same time, (14) is repeated for each observed landmark and the mean and the covariance of the robot pose are updated based on the previously updated one.

2.2. Feature update

The feature update step uses the observations to update the location of the landmark n_t that already registered in the map. A set of sigma points are sampled from the observed feature as:

$$\begin{aligned} \chi^{[0][m]} &= \mu_{n_t,t-1}^{[m]} \\ \chi^{[i][m]} &= \mu_{n_t,t-1}^{[m]} + \left(\sqrt{(n+\lambda)\Sigma_{n_t,t-1}^{[m]}}\right)_i \quad (i = 1, \ldots, n) \\ \chi^{[i][m]} &= \mu_{n_t,t-1}^{[m]} - \left(\sqrt{(n+\lambda)\Sigma_{n_t,t-1}^{[m]}}\right)_i \quad (i = n+1, \ldots, 2n) \end{aligned} \quad (16)$$

where $\mu_{n_t,t-1}^{[m]}$ and $\Sigma_{n_t,t-1}^{[m]}$ are the mean and covariance matrix of the nth feature that is registered in feature initialization step. The predicted measurement $\hat{z}_t^{[m]}$ and Kalman gain $\bar{K}_t^{[m]}$ are calculated as:

$$\begin{aligned} \bar{Z}_t^{[i][m]} &= h\left(\chi^{[i][m]}, s_{t|t}^{[m]}\right) \quad (i = 0, \ldots, 2n) \\ \hat{z}_t^{[m]} &= \sum_{i=0}^{2n} \omega_g^{[i]} \bar{Z}_t^{[i][m]} \end{aligned} \quad (17)$$

$$\begin{aligned} \bar{S}_t^{[m]} &= \sum_{i=0}^{2n} \omega_c^{[i]} \left(\bar{Z}_t^{[i][m]} - \hat{z}_t^{[m]}\right)\left(\bar{Z}_t^{[i][m]} - \hat{z}_t^{[m]}\right)^T + R_t \\ \bar{\Sigma}_t^{[m]} &= \sum_{i=0}^{2n} \omega_c^{[i]} \left(\chi^{[i][m]} - \mu_{n_t,t-1}^{[m]}\right)\left(\bar{Z}_t^{[i][m]} - \hat{z}_t^{[m]}\right)^T \\ \bar{K}_t^{[m]} &= \bar{\Sigma}_t^{[m]} \left(\bar{S}_t^{[m]}\right)^{-1} \end{aligned} \quad (18)$$

Finally, the mean $\mu_{n_t,t}^{[m]}$ and the covariance $\Sigma_{n_t,t}^{[m]}$ of the n_tth feature are updated as follows:

$$\begin{aligned} \mu_{n_t,t}^{[m]} &= \mu_{n_t,t-1}^{[m]} + \bar{K}_t^{[m]} \left(z_t - \hat{z}_t^{[m]}\right) \\ \Sigma_{n_t,t}^{[m]} &= \Sigma_{n_t,t-1}^{[m]} - \bar{K}_t^{[m]} \bar{S}_t^{[m]} \left(\bar{K}_t^{[m]}\right)^T \end{aligned} \quad (19)$$

2.3. Feature initialization

When a new feature is observed for first time, it should be initialized. The feature mean $\mu_{n,t}^{[m]}$ and the feature covariance $\Sigma_{n,t}^{[m]}$ are initialized as a function of the robot pose $s_t^{[m]}$ and measurement z_t as follows [8,9]:

$$\begin{aligned} \psi &= \left[z_t \quad z_t + \sqrt{(n+\lambda)R_t} \quad z_t - \sqrt{(n+\lambda)R_t}\,\right] \\ \bar{M}_t^{[i][m]} &= h^{-1}\left(\psi^{[i][m]}, s_t^{[m]}\right) \quad (i = 0, \ldots, 2l) \\ \mu_{n_t,t}^{[m]} &= \sum_{i=0}^{2l} \omega_g^{[i]} \bar{M}_t^{[i][m]} \\ \Sigma_{n_t,t}^{[m]} &= \sum_{i=0}^{2l} \omega_c^{[i]} \left(\bar{M}_t^{[i][m]} - \mu_{n_t,t}^{[m]}\right)\left(\bar{M}_t^{[i][m]} - \mu_{n_t,t}^{[m]}\right)^T \end{aligned} \quad (20)$$

2.4. Calculation of the importance weight

The importance weight for each particle is computed by:

$$w_t^{[m]} = w_{t-1}^{[m]} \frac{p\left(z_t|s_t^{[m]}\right)p\left(s_t^{[m]}|s_{t-1}^{[m]}, u_t\right)}{N\left(s_t; s_t^{[m]}, P_t^{[m]}\right)} \tag{21}$$

Since the variance of the importance weights increase over time,[21,22] particles are resampled according to their weights. In the resampling process, particles with low weights are eliminated and particles with high weights are multiplied.

3. Intelligent UFastSLAM with MCMC move step

In this section, to increase of consistency, the intelligent UFastSLAM with MCMC move step is presented. A filter is called consistent if its state estimation errors satisfy the following equations [23]:

$$E[s_t - \hat{s}_{t|t}] \triangleq E[e_t] = 0 \tag{22}$$

$$E([s_t - \hat{s}_{t|t}][s_t - \hat{s}_{t|t}]^T) = E[e_t e_t^T] = \hat{P}_{t|t} \tag{23}$$

where $\{\hat{s}_{t|t}, \hat{P}_{t|t}\}$ are the estimated mean and covariance of states at time t, respectively. In above equations, condition (25) is the unbiasedness requirement for estimates, while (26) is covariance matching requirement. Ideally, in order to measure the consistency of a filter, one would compare its estimate with the probability density function (PDF) obtained from an ideal Bayesian filter. This is not practical when PDF is not available. However, if the true state s_t is known, we can use the normalized estimation error squared (NEES) to carry out the consistency test as [4]:

$$\varepsilon_t = (s_t - \hat{s}_{t|t})^T P_{t|t}^{-1}(s_t - \hat{s}_{t|t}) \tag{24}$$

where s_t is the ground truth. The consistency is evaluated by performing multiple Monte Carlo runs and computing the average NEES. Given N_R runs, the average NEES is computed as:

$$\bar{\varepsilon}_t = \frac{1}{N_R}\sum_{i=1}^{N_R} \varepsilon_{it} \tag{25}$$

Considering a consistent linear-Gaussian filter, $N_R\bar{\varepsilon}_t$ has a χ^2 density with $N_R \dim(s_t)$ degrees of freedom. Details about measuring consistency of a filter via the average NEES can be found in [23]. In UFastSLAM, UKF is used to design the proposal distribution and the estimation of features. On the other hand, the consistency of UKF relies on the system model. This model consists of the state equation, the measurement estimation and a priori knowledge of the process and measurement noise statistics (i.e. Q_t and R_t, respectively). This is because the conditional PDF of the state s_t at time t is as:

$$p(s_t|z_{0:t}) = N(s_t;\ \hat{s}_{t|t}, \hat{P}_{t|t}) \tag{26}$$

If this model is completely accurate, Equation (29) holds exactly. Since all models contain some approximations in practice and as, a priori knowledge of the process and measurement noise statistics is unknown in real-life applications, there is a mismatch between estimated PDF with actual PDF. This problem affects the consistency of filter.

In addition, since the variance of the importance weights can increase over time, it is impossible to avoid the degeneracy phenomenon. In this situation, resampling is in need. Although the resampling step reduces the effects of the degeneracy problem, it causes other practical problems. Whenever resampling is performed, 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 on, the number of distinct particles and consequently the covariance of them will decrease, which would lead to inconsistency of the filter. To solve these problems, statistics is tuned using ANFIS and the particle impoverishment induced by resampling is averted with MCMC move step.

3.1. Tuning of statistics using ANFIS

As mentioned, a limitation in applying UFastSLAM to practical application is that a priori knowledge of the process and measurement noise covariance matrices are assumed to be available, which is difficult in real-world problems due to the fact that the noise statistics may change with time. An incorrect a priori knowledge of Q_t and R_t may lead to the performance degradation.[16–20] One of the efficient ways to overcome this weakness is to use an adaptive algorithm. The benefit of the adaptive algorithm is that it keeps the covariance consistent with the real performance. Since features are static, the process noise covariance Q_t is assumed completely known. Hence, the adaptive algorithms to adjust the measurement noise covariance R_t can be derived. In general, the two major approaches to adapt Kalman filter are innovation adaptive estimation (IAE) and multiple model adaptive estimation.[16–20] In this paper, the IAE approach is used to adjust R_t. This technique is known as covariance matching. The basis of this technique is to make the actual value of the covariance of the innovations sequence to be consistent with its theoretical value. From the incoming measurement z_t and the predicted measurement $\hat{z}_t$, the innovation sequence r_t is defined as:

$$r_t = (z_t - \hat{z}_t) \tag{27}$$

where $\hat{z}_t$ is:

$$\hat{z}_t = \sum_{i=0}^{2n} \omega_g^{[i]} \bar{Z}_t^{[i]} \tag{28}$$

The innovation sequence denotes the discrepancy between the predicted and the actual measurement. It represents the additional information available to the filter as a consequence of the new observation z_t. The weighted innovation acts as a correction to the predicted estimate $\mu_{n_{t,t-1}}^{[m]}$ to form the estimation of $\mu_{n_{t,t}}^{[m]}$. The theoretical covariance of innovation sequence S_t is as follows:

$$S_t = \sum_{i=0}^{2n} \omega_c^{[i]} \left(\bar{Z}_t^{[i]} - \hat{z}_t^{[m]}\right)\left(\bar{Z}_t^{[i]} - \hat{z}_t\right)^T + R_t \tag{29}$$

where $\bar{Z}_t^{[i]}$ is:

$$\bar{Z}_t^{[i]} = h(\chi^{[i]}, s_t) \tag{30}$$

The actual covariance of innovation sequence $\hat{C}_k$ is approximated by the sample covariance through averaging a moving estimation window of size N_w as follows:

$$\hat{C}_t = \frac{1}{N} \sum_{i=i_0}^{t} (r_i^T r_i) \tag{31}$$

where $i_o = t - N_w + 1$ represents the first sample of the estimation window. The window size N_w is chosen empirically to give some statistical smoothing. To minimize the discrepancy between the actual covariance of the innovation sequence and the theoretical value, the covariance R_t is tuned to correct it. For this purpose, a new variable DOM_t called a degree of matching (DOM) is defined as:

$$\text{DOM}_t = S_t - \hat{C}_t \tag{32}$$

The variable DOM_t indicates the extent of discrepancy between the actual value and the theoretical one. The logic of the adaptation algorithm using covariance matching technique can be described as follows: if the actual covariance value $\hat{C}_t$ is observed, whose value is within the range predicted by theory S_t, and the difference is very near to zero, this indicates that both covariance matrices match almost perfectly and only a small change is needed to be made on the value of R_t. If the actual covariance is less than that of theoretical value, the value of R_t should be decreased. On the contrary, if $\hat{C}_t$ is greater than that of theoretical value; the value of R_t should be increased. Thus, R_t can be used to vary S_t in accordance with the value of DOM_t in order to reduce the discrepancies between S_t and $\hat{C}_t$. The adaptation of the (i, i)th element of R_t is made in accordance with the (i, i)th element of DOM_t. This adjustment mechanism lends itself very well to being dealt with using a fuzzy-logic approach.

In this paper, ANFIS is proposed to adjust R_t. Since the size of DOM_t and R_t are both $2z_f \times 2z_f$, where z_f is the number of reobserved features, the overall ANFIS employs a bank of subsystems where each subsystem is a two-input-single-output ANFIS. These ANFISs are employed to tune each diagonal of the element of R_t. The ANFIS is a five layers network as shown in Figure 1. The inputs of ANFIS are DOM_t and ∇DOM_t in which ∇DOM_t is defined as:

$$\nabla\text{DOM}_t = \text{DOM}_t - \text{DOM}_{t-1} \tag{33}$$

The covariance matrix R_t is tuned by ANFIS as:

$$R_t = R_t + \nabla R_t \tag{34}$$

where ∇R_t is ANFIS output variable. For a clear understanding of the structure of ANFIS, let us denote the input and output of the ith node in the ℓth layer by $u_i^{(\ell)}$ and $f_i^{(\ell)}$, respectively. The functions of nodes from layer one to layer five of ANFIS are as follows:

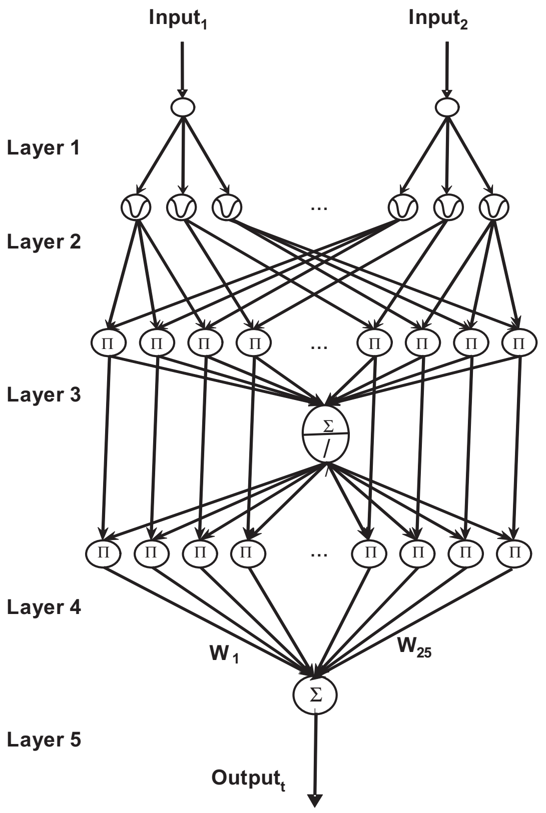

Figure 1. The ANFIS structure for tuning R_t.

Layer 1: The nodes in this layer only transmit input values to the next layer as:

$$f_i^{(1)} = u_i^{(1)} \tag{35}$$

Layer 2: The nodes in this layer represent Gaussian membership functions with the mean m and the standard deviation σ. The output of the ith membership function is as:

$$f_{ij}^{(2)} = \exp\left[-\frac{1}{2}\left(\frac{u_{ij}^{(2)} - m_{ij}}{\sigma_{ij}}\right)^2\right] \tag{36}$$

where subscript ij indicates the jth term of the ith input.

Layer 3: The nodes in this layer are rule nodes and constitute the antecedents of fuzzy rule base. Each node is equivalent to a cluster in the input space. A rule node performs a fuzzy AND operation (or product inference) for calculating the firing strength. Thus, the input and output functions of the ℓth rule node are:

$$f_\ell^{(3)} = \prod_i u_i^{(3)} \tag{37}$$

Layer 4: The node in this layer performs the normalization of firing strengths from layer three as:

$$f_\ell^{(4)} = \frac{u_\ell^{(4)}}{\sum_{\ell=1}^{25} u_\ell^{(4)}} \tag{38}$$

Layer 5: This layer is the output layer. The link weights in this layer represent the singleton constituents w_ℓ of the output variable. The output node integrates all the normalization firing strengths from layer four with the corresponding singleton constituents and acts as a defuzzfier:

$$\nabla R_t = \sum_{\ell=1}^{25} f_\ell^{(4)} w_\ell \tag{39}$$

The fuzzy rules to tune R_t are represented in Table 1. Also, the membership functions for inputs and output of these rules are shown in Figures 2 and 3. The training algorithm is to adjust the network weights through the minimization of the following cast function:

Table 1. Rule table.

		Input$_2$				
Input$_1$		L	LM	Z	HM	H
	L	H	H	MH	ZH	Z
	LM	H	MH	ZH	Z	ZL
	Z	MH	ZH	Z	ZL	ML
	HM	ZH	Z	ZL	ML	L
	H	Z	ZL	ML	L	L

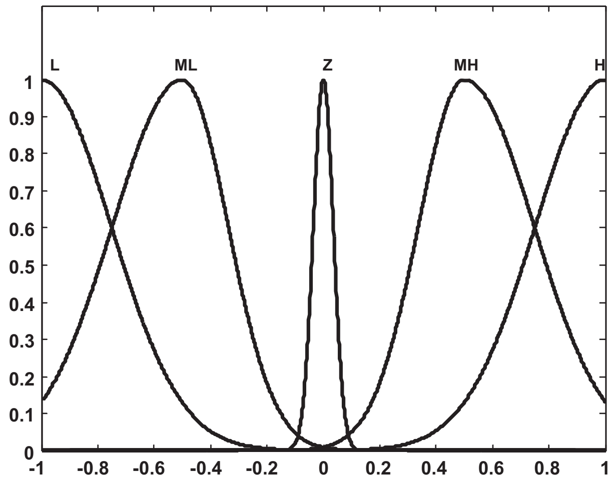

Figure 2. Inputs membership functions.

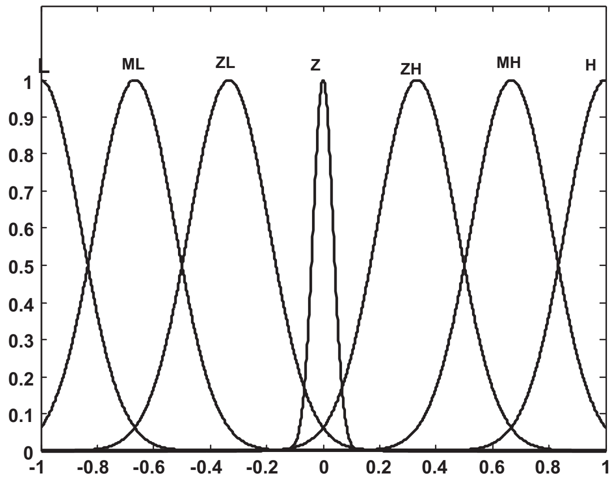

Figure 3. Output membership function.

$$E = \frac{1}{2} e_t^2 \tag{40}$$

where

$$e_t = S_t - \hat{C}_t \tag{41}$$

The back-propagation learning algorithm is employed to tune the weighting vector of ANFIS as follows:

$$W_{t+1} = W_t - \eta \frac{\partial E}{\partial W_t} \tag{42}$$

where η is the learning rate and $W = [m, \sigma, w]^T$ is the weighting vector of ANFIS, respectively. The gradient of E with respect to W is as:

$$\frac{\partial E}{\partial W_t} = -e_t \frac{\partial \nabla R_t}{\partial W_t} \tag{43}$$

As a result, the weighting vector of ANFIS is tuned as:

$$W_{t+1} = W_t + \eta e_t \frac{\partial \nabla R_t}{\partial W_t} \quad (44)$$

3.2. MCMC move step

In general, a common problem in the particle filtering and consequently UFastSLAM is that the variance of importance weights increases over time. Hence, a large computational effort is wasted on updating the particles with negligible weights. To solve this problem, resampling is performed, whereby samples are chosen with replacement from the original sample set, to generate an equal-weight sample set. However, resampling step leads to a loss of diversity among the particles and consequently filter inconsistency. The loss of diversity is due to the fact that in resampling step, samples are drawn from a discrete distribution rather than a continuous one. To overcome this problem, it is important to determine when and how the resampling step must be performed. Liu introduced effective number of particles N_{eff} to estimate how well the current particle set represents the true posterior and it is calculated by Stachniss [24]:

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{M} (w_t^{[i]})^2} \quad (45)$$

where $w_t^{[i]}$ refers to the normalized weight of particle i and M is the number of particles. The resampling process is operated whenever N_{eff} is below a predefined threshold N_{tf}. In this paper, the above criterion is used for resampling. In the field of PFs, many algorithms have been researched for performing resampling. The most representative resampling algorithms are multinomial resampling and stratified resampling.[25] In UFastSLAM, the common algorithm is residual stratified resampling (RSR), which uses weight strata to decide how many copies of each particle should be made. Although the RSR technique has nice performance, it causes the loss of particle diversity. In order to restrain losing diversity, we require a procedure to introduce sample variety after the resampling step without affecting the validity of the approximation. In this paper, the MCMC method is used to increase the diversity of particles and to improve the performance of filtering. The MCMC methods construct a Markov chain for each sample after the resampling step without affecting the posterior estimation. To describe MCMC method, assume that particles are distributed according to a posterior $p(\tilde{x}_{0:t}|z_{1:t})$ and then applying a Markov chain transition kernel $\kappa(x_{0:t}|\tilde{x}_{0:t})$ with invariant distribution $p(x_{0:t}|y_{1:t})$ such that:

$$\int \kappa(x_{0:t}|\tilde{x}_{0:t}) p(\tilde{x}_{0:t}|y_{0:t}) = p(x_{0:t}|y_{0:t}) \quad (46)$$

We still have a set of particles distributed according to the posterior of interest $p(\tilde{x}_{0:t}|z_{1:t})$. However, the new particles might have been moved to areas of high likelihood, and the total variance of current distribution with respect to the invariant distribution can only decrease. The Metropolis-Hastings (MH) algorithm [26] is one of the most common MCMC methods. The MH algorithm employs a conditional density, also known as the proposal distribution, to generate Markov chain with an invariant distribution. The standard MH algorithm is summarized as follows [27]:

```
1. sample $u \sim U_{[0,1]}$, $U_{[0,1]}$ is uniformly distribution in the
   interval $[0,1]$
2. sample the proposal candidate $x_t^{*[i]} \sim p(x_t|x_{t-1}^{[i]})$
3. if $u \le \min\left\{1, \frac{p(y_t|x_t^{*[i]})}{p(y_t|\tilde{x}_t^{[i]})}\right\}$
        accept move:
        $x_{0:t}^{(i)} = (\tilde{x}_{0:t-1}^{(i)}, x_t^{*(i)})$
     else
        reject move:
        $x_{0:t}^{(i)} = \tilde{x}_{0:t}^{(i)}$
   end if
```

In the proposed method, the MH algorithm is used on each particle after resampling $\tilde{s}^{t[m]}$ using a Markov transition kernel with invariant distribution given by $p\left(s^{t,[m]}|z^t, u^t, n^t\right)$ to obtain $\left(s^{t,[m]}, P^{t,[m]}\right)$ as follows:

1. Sample u from a uniform distribution: $u \sim U_{[0,1]}$

2. Predict particles with UKF

a. Calculation of sigma points for time update

The augmented state and augmented covariance matrix by appending the mean and covariance of process noise vector as follows:

$$\tilde{s}^{a[m]}_{t-1|t-1} = \begin{bmatrix} \tilde{s}^{[m]}_{t-1|t-1} \\ 0 \end{bmatrix} \qquad \tilde{P}^{a}_{t-1} = \begin{bmatrix} \tilde{P}_{t-1|t-1} & 0 \\ 0 & Q_t \end{bmatrix}$$

where $\tilde{s}^{a}_{t-1|t-1}$ and $\tilde{P}^{a}_{t-1}$ are the augmented state and covariance matrix, respectively. To predict over a time step $t-1$ to t, the sigma points is defined as:

$$\tilde{\delta}^{a[0][m]}_{t-1|t-1} = \tilde{s}^{a[0][m]}_{t-1|t-1}$$
$$\tilde{\delta}^{a[i][m]}_{t-1|t-1} = \tilde{s}^{a[i][m]}_{t-1|t-1} + \left(\sqrt{(n+\lambda)\tilde{P}^{a[m]}_{t-1|t-1}}\right)_i \quad i = 1,\ldots,n$$
$$\tilde{\delta}^{a[i][m]}_{t-1|t-1} = \tilde{s}^{a[i][m]}_{t-1|t-1} - \left(\sqrt{(n+\lambda)\tilde{P}^{a[m]}_{t-1|t-1}}\right)_i \quad i = n+1,\ldots,2n$$
$$\tilde{\delta}^{a[i]}_{t-1|t-1} = \begin{bmatrix} \tilde{\delta}^{[i][m]}_{t-1|t-1} & \delta^{u[i][m]}_{t} \end{bmatrix}^T$$

b. Time update

The mean and covariance of particles is predicted using motion model as follows:

$$\delta^{*[i][m]}_{t|t-1} = f(\tilde{\delta}^{[i][m]}_{t-1|t-1}, u_t + \delta^{u[i][m]}_t)$$
$$s^{*[m]}_{t|t-1} = \sum_{i=0}^{2n} \omega^{[i]}_g \delta^{*[i][m]}_{t|t-1}$$
$$P^{*[m]}_{t|t-1} = \sum_{i=0}^{2n} \omega^{[i]}_c (s^{*[m]}_{t|t-1} - \delta^{*[i][m]}_{t|t-1})(s^{*[m]}_{t|t-1} - \delta^{*[i][m]}_{t|t-1})^T$$

3. Update particles with UKF

By incorporating new observation, the mean of the particles is updated as follows:

$$\zeta^{[i][m]}_t = h(\delta^{*[i][m]}_{t|t-1}, \mu^{[m]}_{k,t-1})$$
$$z^{*[m]}_t = \sum_{i=0}^{2n} w^{[i]}_g \zeta^{[i][m]}_t$$
$$S^{*[m]}_t = \sum_{i=0}^{2n} w^{[i]}_c (\zeta^{[i][m]}_t - z^{*[m]}_t)(\zeta^{[i][m]}_t - z^{*[m]}_t)^T + R_t$$
$$P^{*[m]}_{\delta\upsilon} = \sum_{i=0}^{2n} \omega^{[i]}_c (s^{*[m]}_{t|t-1} - \delta^{*[i][m]}_{t|t-1})(\zeta^{[i][m]}_t - z^{*[m]}_t)^T$$
$$s^{*[m]}_{t|t} = s^{*[m]}_{t|t-1} + K^{*[m]}_t (z_t - z^{*[m]}_t)$$

where gain $K^{*[m]}_t$ is as:

$$K^{*[m]}_t = P^{*[m]}_{\delta\upsilon} (S^{*[m]}_t)^{-1}$$

The covariance of the particles is updated as follows:

$$P^{*[m]}_{t|t} = P^{[m]}_{t|t-1} - K^{*[m]}_t S^{*[m]}_t (K^{*[m]}_t)^T$$

4. Sample from the proposal candidate

$$s^{*[m]}_t \sim q(s_t|\tilde{s}^{t-1,[m]}, z^t, u^t, n^t) = N(s^*_t; s^{*[m]}_{t|t}, \hat{P}^{*[m]}_{t|t})$$

5. Calculate acceptance probability

When the proposal distribution is chosen as $q(s_t|\tilde{s}^{t-1,[m]}, z^t, u^t, n^t)$, the acceptance probability is:

$$\gamma = \min\left\{1, \frac{p(z_t|s^{*t,[m]}, z^{t-1}, u^t, n^t)p(s^{*[m]}_t|\tilde{s}^{[m]}_{t-1}, u_t)q(\tilde{s}^{[m]}_t|\tilde{s}^{t-1,[m]}, z^t, u^t, n^t)}{p(z_t|\tilde{s}^{t,[m]}, z^{t-1}, u^t, n^t)p(\tilde{s}^{[m]}_t|\tilde{s}^{[m]}_{t-1}, u_t)q(s^{*[m]}_t|\tilde{s}^{t-1,[m]}, z^t, u^t, n^t)}\right\}$$

6. Accept or reject

```
    **If** $u \leqslant \gamma$
        Accept move with probability $\gamma$ as:
            $s^{t,[m]} = (\tilde{s}^{t-1,[m]}, s^{*,[m]}_t)$
            $P^{t,[m]} = (\tilde{P}^{t-1,[m]}, P^{*,[m]}_t)$
    **else**
        Reject move with probability $1 - \gamma$ as:
            $s^{t,[m]} = \tilde{s}^{t,[m]}$
            $P^{t,[m]} = \tilde{P}^{t,[m]}$
    **end if**
```

With this algorithm, we see that MCMC move step accepts a move with probability γ. Therefore, the acceptance probability increases as the probability of the new particle increases, and the old particle is changed to a new one, if the old one has a lower PDF.

4. Results

4.1. Simulation results

To verify the effectiveness of the proposed approach in comparison with UFastSLAM, simulation experiments have been carried out. The proposed algorithm for the SLAM problem has been tested for the benchmark environment, with varied number and position of the features, available in [28]. The simulation environment for testing is shown in Figure 4, where the star points ($*$) indicate landmarks, and the curve shows the path of the mobile robot.

The number of landmarks is 35, and they are unknown for the mobile robot. The initial position of the robot is assumed to be $s_0 = (0, 0, 0)$, and the robot moves at a speed 3 m/s and with a maximum steering angle of 30°. The robot has 4 m wheel base and is equipped with a range-bearing sensor with a maximum range of 30 m and 180° frontal field-of-view. The control noise is $\sigma_v = 0.3$ m/s, $\sigma_\gamma = 3°$, and the measurement noise is 0.1 m in range and 0.1° in bearing. A control frequency is 40 Hz and observation scans are obtained at 5 Hz.

At first, we compare the performance of the two algorithms while the measurement noise is $\sigma_r = 0.1$ m, $\sigma_\theta = 1°$ and the control noise is $\sigma_r = 0.3$ m/s, $\sigma_\theta = 3°$. Figures 5 and 6 show results for this case. Since the MCMC move step increases the diversity of particles, the proposed method achieves better precision than UFast-SLAM. To verify the consistency of both algorithms, average NEES is used as a measure factor. For the

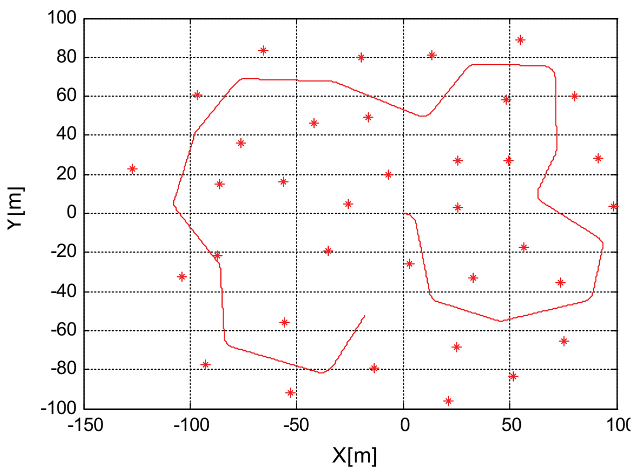

Figure 4. The experiment environment: true robot path and true landmarks (*).

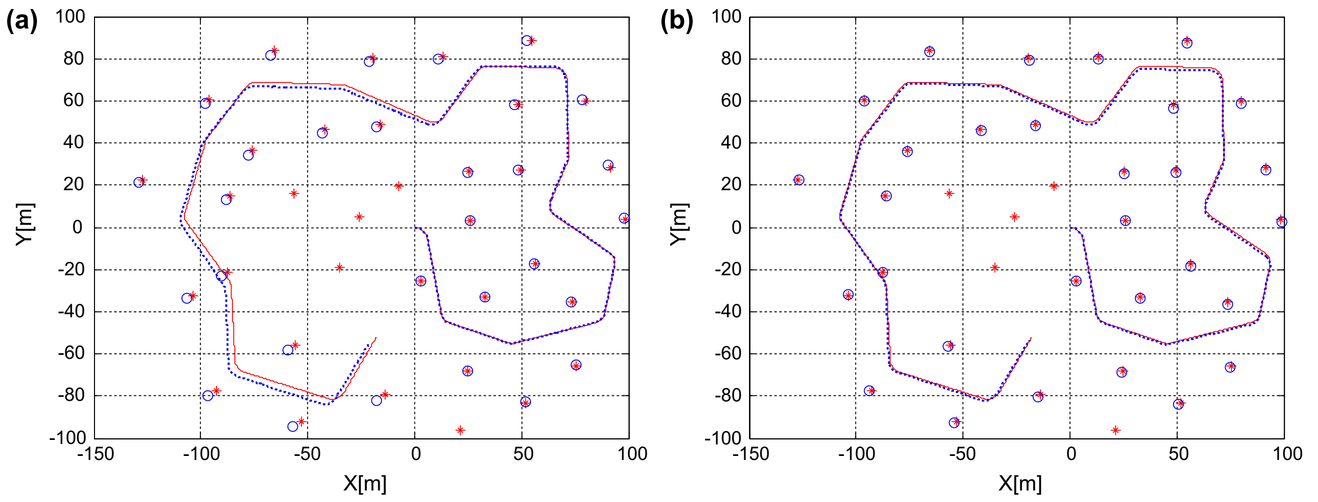

Figure 5. Estimated robot path and the estimated landmark with true robot path and true landmarks. The '…' is the estimated path, the 'o' are the estimated landmark positions: (a) UFastSLAM (b) proposed method.

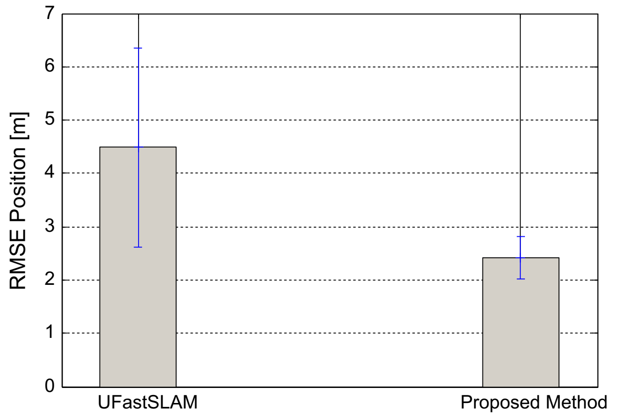

Figure 6. Root mean square error (RMSE) of the robot position.

2-dimensional vehicle position, with twenty Monte Carlo simulations, the two-sided 95% probability concentration region for $\bar{\varepsilon}_k$ is bounded by interval [1.3, 2.79]. Figure 7 shows that the proposed method stays consistent, whereas the NEES of UFastSLAM soars after 30 s.

An estimate of the rate of loss of particle diversity is obtained by recording the number of distinct particles in the set representing a landmark. Figure 8 shows that the number of distinct particles in the proposed method is more than that of UFastSLAM. Therefore, the consistency of the proposed method is more than that of UFastSLAM.

Next, the performance of algorithms is compared with various numbers of particles in Figure 9. It is observed that the proposed method needs a lower number of particles to obtain the same estimation accuracy as that of UFastSLAM.

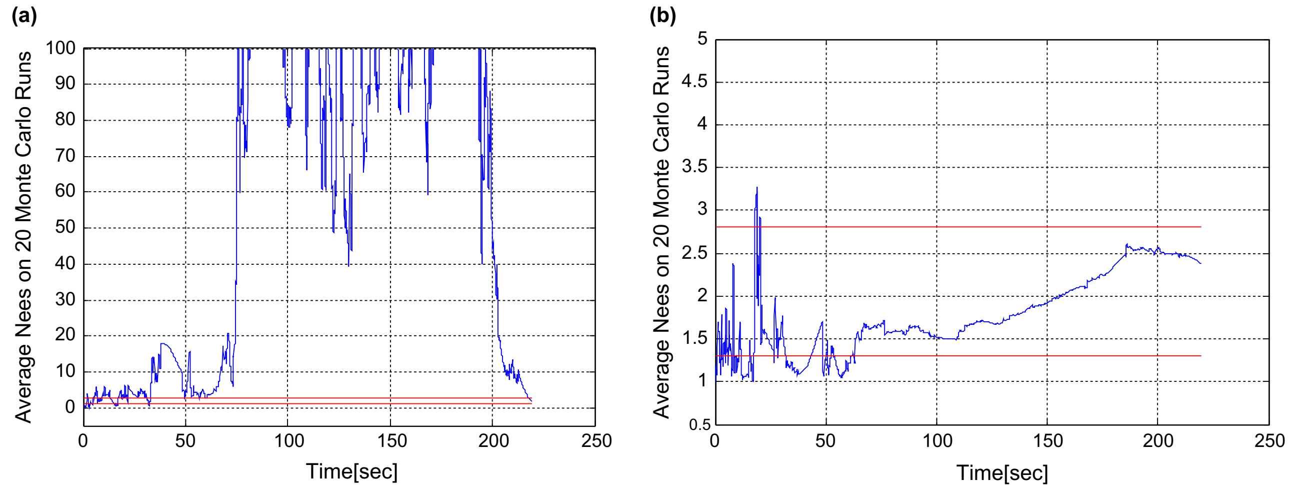

Figure 7. Consistency: (a) UFastSLAM and (b) proposed method.

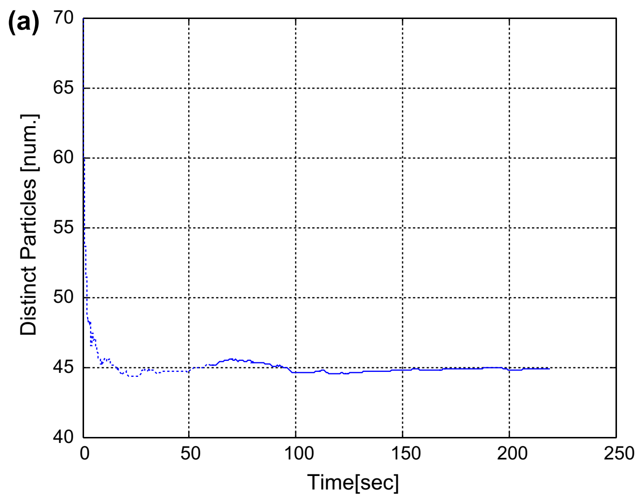

Figure 8. Number of distinct particles (a) UFastSLAM (b) proposed method.

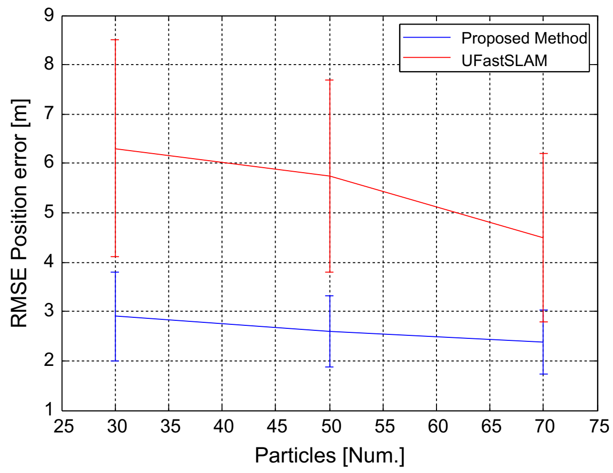

Figure 9. Performance of the proposed method and UFastSLAM with different numbers of particles.

Finally, we compare the performance of the two algorithms while varying the level of measurement noise wrongly as $\sigma_r = 2\text{m}$, $\sigma_\theta = 0.5^\circ$ and fixing the control noise as $\sigma_v = 0.3\text{m/s}$, $\sigma_\gamma = 3^\circ$. The performance of the proposed method is compared with UFastSLAM where its measurement covariance matrix R_t is kept static throughout the experiment. The proposed algorithm starts with a wrongly known statistics and then adapts the R_t through ANFIS and attempts to minimize the mismatch between the theoretical and actual values of the innovation sequence. The free parameters of ANFIS are automatically learned by gradient descent method during training. Figures 10 and 11 show the comparison between the proposed algorithm and UFastSLAM. It can be clearly seen that the results of the proposed algorithm are better than that of UFastSLAM. In the proposed algorithm, the estimated vehicle path and estimated

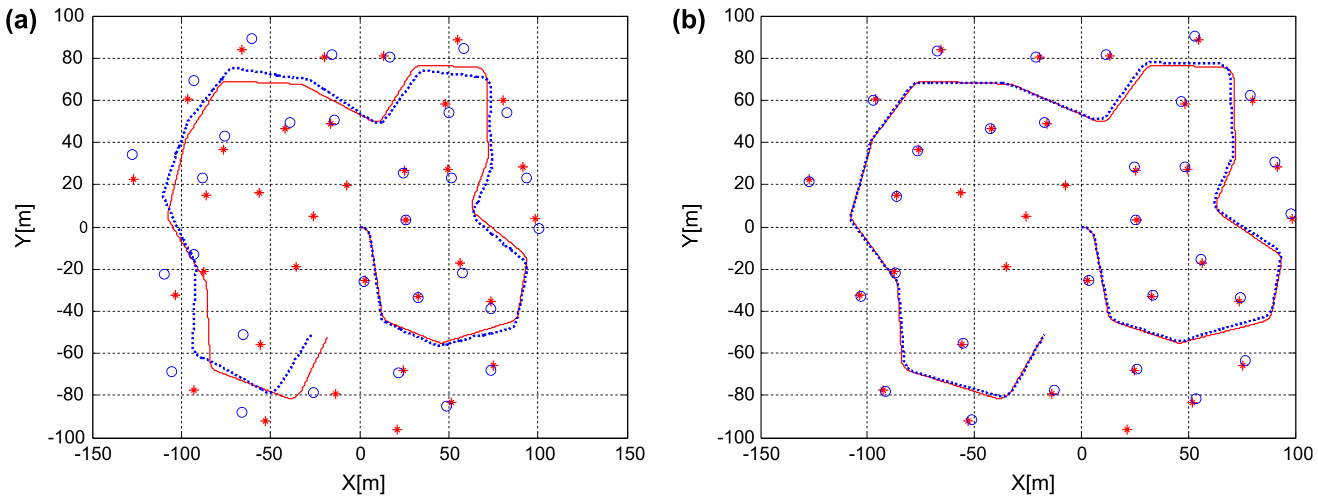

Figure 10. Estimated robot path and the estimated landmark with true robot path and true landmarks. The '…' is the estimated path, the 'o' are the estimated landmark positions: (a) UFastSLAM and (b) proposed method.

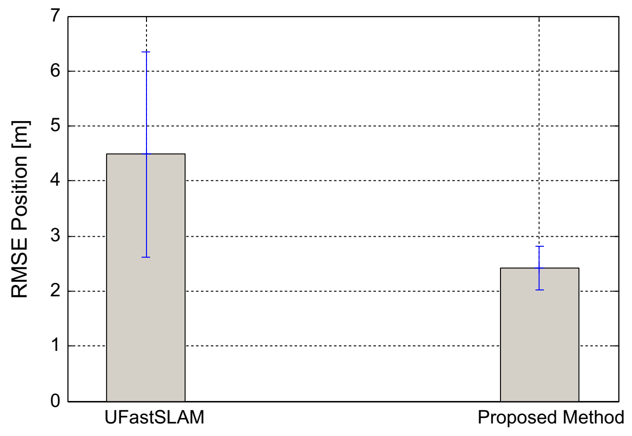

Figure 11. RMSE of the robot position.

landmark coincide as closely as possible with the actual trajectory and the actual positions features. This is because the proposed method adaptively tuned the measurement covariance matrix R_t. In fact, the matrix R_t converges to its actual value while in UFastSLAM matrix R_t is kept fixed over time as shown in Figure 12. Also, as shown in Figure 13, the consistency of the proposed method is more than that of UFastSLAM in this situation.

4.2. *Experimental results*

We have evaluated the proposed algorithm on the car park data set and the Sydney Victoria Park data set, two popular data sets in the SLAM community.[29]

The first experimental is done on the car park data set. The vehicle is driven around the park. The velocity and the steering angle are measured with encoders, but uneven terrain induced additional nonsystematic errors because of wheel slippage and vehicle attitude. Consequently, the odometry information from the encoder is poor as shown in Figure 14. In this experiment, artificial

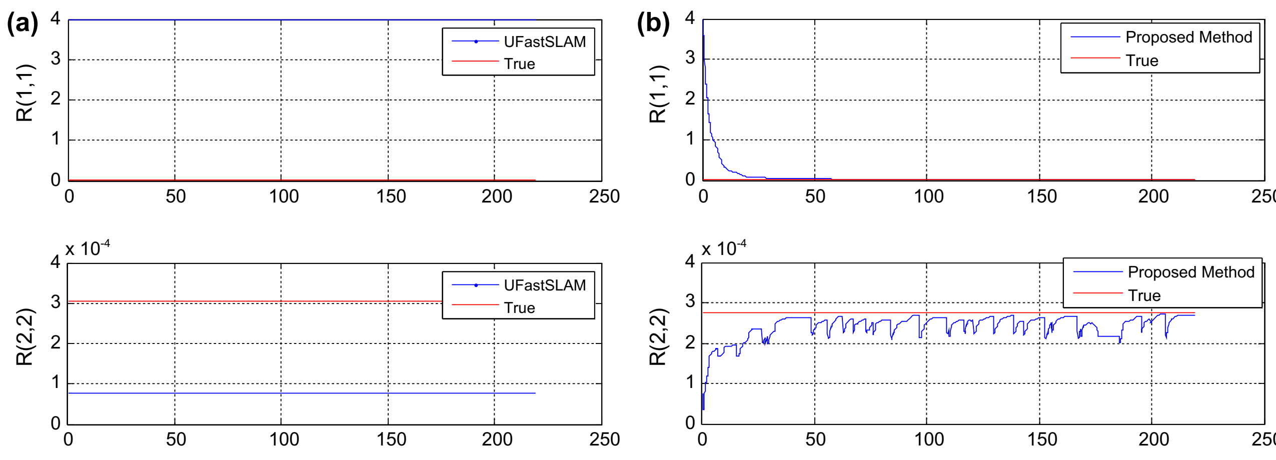

Figure 12. (a) UFastSLAM (b) proposed method.

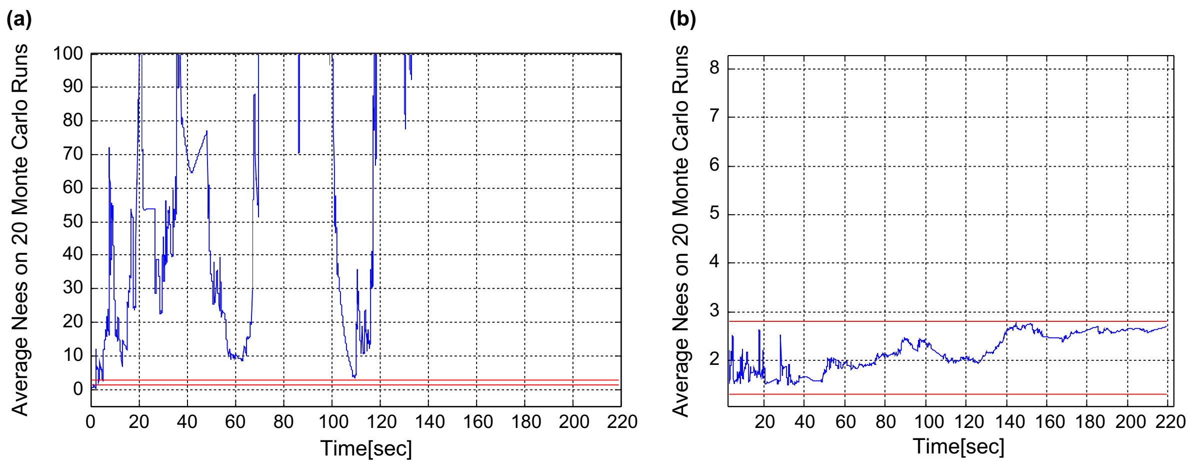

Figure 13. Consistency: (a) UFastSLAM and (b) proposed method.

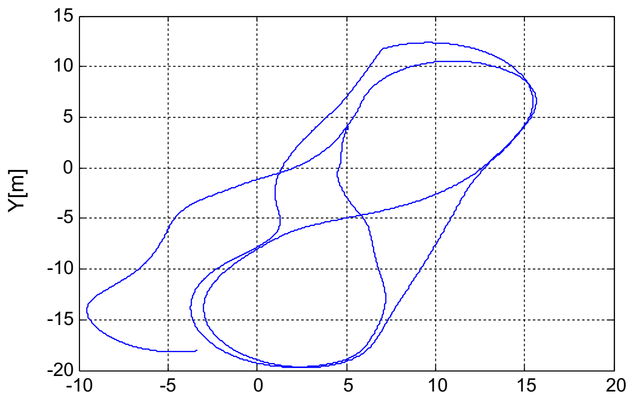

Figure 14. Odometry of the vehicle.

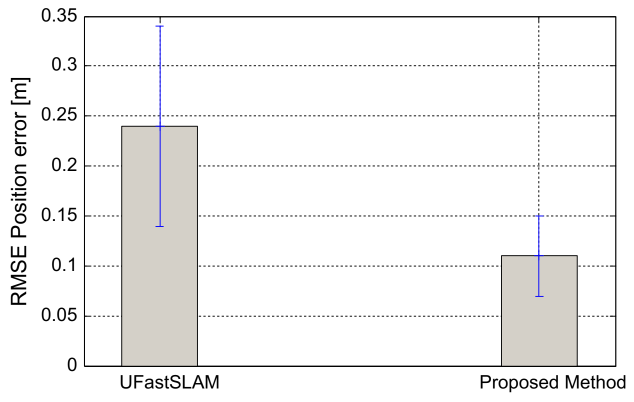

Figure 16. RMSE of the robot position.

features are used. Since the true position of the features is obtained with a global positioning system (GPS), a true navigation map is available. Moreover, a GPS receiver is used to provide ground truth for the robot position. The performance of the proposed algorithm is compared with UFastSLAM in situation that the correspondences between the observation and the features are assumed to be unknown and 20 particles are used for two algorithms. Each algorithm was run many times to confirm the variance of the estimate error. For the unknown data association, the individual compatibility nearest neighbor test with 2σ acceptance region is used. The proposed algorithm starts with a wrongly known R_t and then adapts the R_t through ANFIS and attempts to minimize the mismatch between the theoretical and actual values of the innovation sequence. Figures 15 and 16 show the comparison between the proposed algorithm and UFastSLAM. Since the proposed algorithm tunes adaptively R_t, the performance of the proposed algorithm is better than that of UFastSLAM. This also improves data association and consistency.

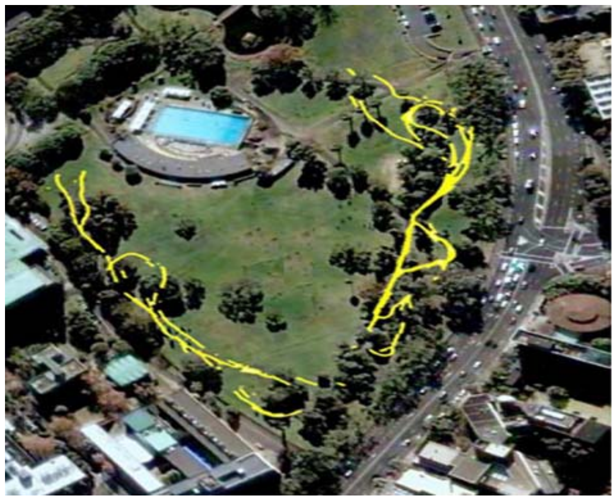

Figure 17. Victoria Park with the GPS path [29]. The yellow path is the GPS path of the robot.

The second experimental is done on Victoria data set that is a larger area with mild uneven terrain and different types of surfaces. Figure 17 shows Victoria Park with

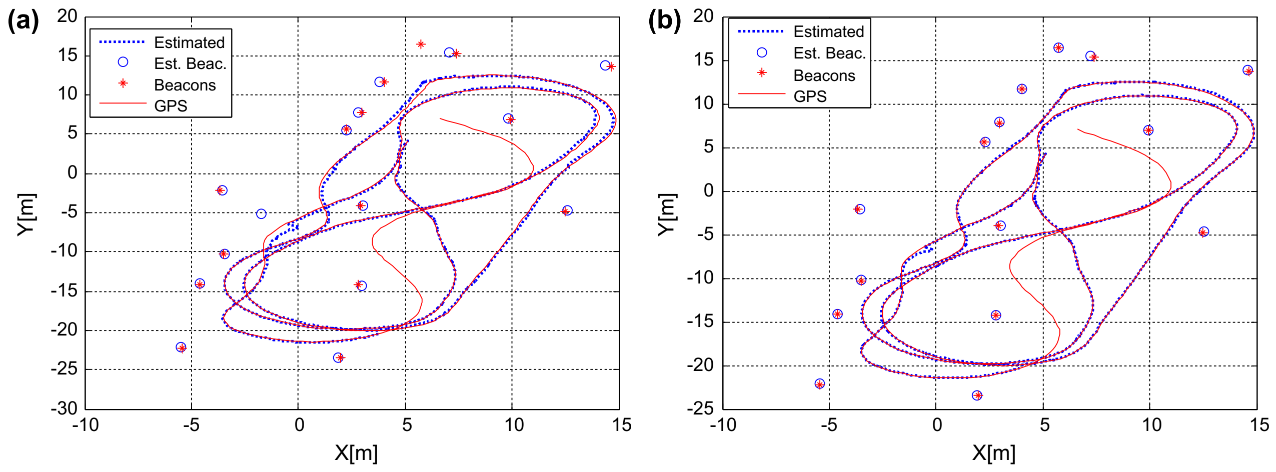

Figure 15. (a) UFastSLAM and (b) proposed method.

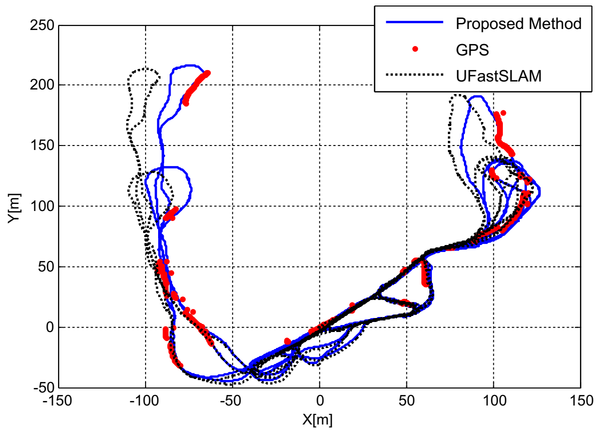

Figure 18. Estimated trajectory by UFastSLAM and the proposed method.

the GPS path. The trajectories for each algorithm are presented in Figure 18. Since the estimated trajectory of the proposed method coincides very well with the GPS path, the performance of the proposed method is better than that of UFastSLAM.

5. Conclusion

The FastSLAM algorithm has two drawbacks, namely the linear approximation of nonlinear functions and the calculation of the Jacobian matrices. UFastSLAM has been recently proposed for solving these problems. However, UFastSLAM is inconsistent because of losing particle diversity and incorrect a priori knowledge of process and measurement noise. In this paper, to enhance consistency an intelligent UFastSLAM with MCMC move step is proposed. In the proposed algorithm, the performance of UFastSLAM is supervised by using ANFIS, and the diversity of particles is increased using MCMC move step. The performance of the proposed algorithm is compared with UFastSLAM for benchmark environment. The results show the effectiveness of the proposed method.

Notes on contributors

Ramazan Havangi received the M.Sc. degrees 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.

Mohammad Ali Nekoui was born in December 1952. He received his M.Sc. degree in Electrical Engineering from the University of Tehran in 1976, Diplome d'Especialisation in Instrumentation et Metrologie from Ecole Superieur d'Electricite (SUPEL EC), France, in 1979 and his Ph.D. degree at the School of Electrical and Electronic Engineering in Computer and Control Department from University of Leeds, UK in 1997. Since 1980, he has been with the K.N. Toosi University of Technology. At present he is an Assistant Professor at the Faculty of Electrical and Computer Engineering of this university. His interest includes linear and nonlinear optimization, linear systems, optimal control, and different aspects of mathematics in control.

Hamid D. Taghirad is currently a Professor with the Faculty of Electrical and Computer Engineering, Department of Systems and Control and the founder and director of the Advanced Robotics and Automated System (ARAS) Research Center at K.N. Toosi University of Technology, Tehran, Iran. He has received his B.Sc. degree in mechanical engineering from Sharif University of Technology, Tehran, Iran, in 1989, his M.Sc. in mechanical engineering in 1993, and his Ph.D. in electrical engineering in 1997, both from McGill University, Montreal, Canada. He has been awarded two visiting professorship positions at McGill University in 2006, and at ETS, Quebec University in 2010. His field of research includes robust and nonlinear control, and applied robotics systems. He has been served as the founder and member of the board of Iranian Society of Mechatronics (ISM), and Iranian Robotics Society (IRS), editor in chief of Mechatronics Magazine, and editorial board of International Journal of Robotics: Theory and Application. He has also served as an organizing committee member of many international conferences in the field of robotics, including International Conference on Robotics and Mechatronics (ICRoM). His publications include five books, and more than 190 papers in peer-reviewed international journals and conference proceedings.

Mohammad Teshnehlab received the B.Sc. degree in electrical engineering from Stony Brook University, Stony Brook, NY, in 1980, the M.S. degree in electrical engineering from Oita University, Oita, Japan, in 1991, and the Ph.D. degree in computational intelligence from Saga University, Saga, Japan, in 1994. 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. He is the author or a coauthor of more than 130 papers that have appeared in various journals and conference proceedings. 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.

References

[1] Thrun S, Montemerlo M, Koller D, Wegbreit B, Nieto J, Nebot E. FastSLAM: an efficient solution to the simultaneous Localization and mapping problem with unknown data association. J. Mach. Learn. Res. 2004;4:380–407.

[2] Chen H, Samarabandu J, Rodringo R. Recent advances in simultaneous localization and map-building using computer vision. Adv. Robot. 2007;21:233–265.

[3] Montemerlo M. FastSLAM: a factored solution to the simultaneous localization and mapping problem with unknown data association [Ph.D. dissertation]. Pittsburgh (PA): Carnegie Mellon University; 2003.

[4] Bailey T, Nieto J, Nebot E. Consistency of the FastSLAM algorithm. In: Proceedings IEEE International Conference on Robotics and Automation; 2006; Orlando, FL. p. 424–429.

[5] Kwak N, Kim IK, Lee HC, Lee BH. Adaptive prior boosting technique for the efficient sample size in FastSLAM. In: Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems; San Diego (CA); 2007. p. 630–635.

[6] Kwak N, Kim G, Lee B. A new compensation technique based on analysis of resampling process in FastSLAM. J. Robot. 2007;26:205–217.

[7] Kim I, Kwak N, Lee H, Lee B. Improved particle fusing geometric relation between particles in FastSLAM. J. Robot. 2009;27:853–859.

[8] Kim C, Sakthivel R, Chung WK. Unscented FastSLAM: a robust algorithm for the simultaneous localization and mapping problem. In: Proceedings IEEE International Conference on Robotics and Automation; 2007; Roma, Italy. p. 2439–2445.

[9] Kim C, Sakthivel R, Chung WK. Unscented FastSLAM: a robust and efficient solution to the SLAM problem. IEEE Trans. Robot. 2008;24:808–820.

[10] Cugliari M, Martinelli F. A FastSLAM algorithm based on the unscented filtering with adaptive selective resampling. In: Proceedings International Conference on Field and Service Robotics; 2008; Chamonix, France. p. 359–368.

[11] Kim C, Kim H, Chung WK. Exactly Rao-Blackwellized unscented particle filters for SLAM. In: Proceedings International Conference on Robotics and Automation; 2011; Shanghai, China. p. 9–13.

[12] Wang X, Zhang H. A UPF-UKF framework for SLAM. In: Proceedings IEEE International Conference on Robotics and Automation; 2007; Roma, Italy. p. 1664–1669.

[13] Li M, Bing-rong H, Rong-hua L, Zhen-Hua W. A novel method for mobile robot simultaneous localization and mapping. J. Zhejiang Univ. Sci. A. 2006;7:937–944.

[14] Julier S, Uhlmann J, Durrant-Whyte HF. A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Trans. Autom. Control. 2000;45:477–482.

[15] Wan E, Van Der Merwe R. The unscented Kalman filter for nonlinear estimation. In: Proceedings Adaptive Systems for Signal Processing, Communications, and Control Symposium; 2000; Lake Louise, Alberta, Canada. p. 153–158.

[16] Fitzgerald RJ. Divergence of the Kalman filter. IEEE Trans. Autom. Control. 1971;16:736–747.

[17] Mehra RK. On the identification of variances and adaptive Kalman filtering. IEEE Trans. Autom. Control. 1970;15:175–184.

[18] Shi Y, Han C, Liang Y. Adaptive UKF for target tracking with unknown process noise statistics. In: Proceedings International Conference on Information Fusion; Seattle (WA); 2009.

[19] Jiang Z, Song Q, He Y, Han J. A novel adaptive unscented Kalman filter for nonlinear estimation. In: Proceedings IEEE Conference on Decision and Control; New Orleans (LA); 2007.

[20] Song Q, Qi J, Han J. An adaptive UKF algorithm and its application in mobile robot control. In: Proceedings IEEE International Conference on Robotics and Biomimetics; 2006; Kunming, China.

[21] Grisetti G, Stachniss C, Burgard W. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposal and selective resampling. In: Proceedings IEEE International Conference on Robotics and Automation; 2005; Barcelona, Spain. p. 2443–2448.

[22] Grisetti G, Stachniss C, Burgard W. Improved techniques for grid mapping with Rao-Blackwellized particle filters. IEEE Trans. Robot. 2007;23:34–46.

[23] Bar-Shalom Y, Li XR, Kirubarajan T. Estimation with applications to tracking and navigation. New York: John Wiley and Sons; 2001.

[24] Stachniss C, Haehnel D, Burgard W, Grisetti G. On actively closing loops in grid-based FastSLAM. Adv. Robot. 2005;19:1059–1080.

[25] Yu JX, Cai ZX, Duan ZH. Survey on some key technologies of mobile robot localization based on particle filter. J. Appl. Res. Comput. 2007;24:9–14.

[26] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller A, Teller H. Equations of state calculations by fast computing machines. J. Chem. Phys. 1953;21:1087–1091.

[27] Andrieu C, De Freitas N, Doucet A, Jordan MI. An introduction to MCMC for machine learning. J. Mach. Learn. 2003;50:5–43.

[28] Online Available from: http://www.personal.acfr.usyd.edu.au/tbailey/software/index.html

[29] Nebot E. Victoria Park data set. Available from: http://www-personal.acfr.usyd.edu.au/nebot/dataset.html.