TÜBİTAK

Turk J Elec Eng & Comp Sci, Vol.20, No.1, 2012, © TÜBİTAK
doi:10.3906/elk-1004-504

An improved FastSLAM framework using soft computing

Ramazan HAVANGI*, Mohammad Ali NEKOUI, Mohammad TESHNEHLAB
Faculty of Electrical Engineering, K.N. Toosi University of Technology, Tehran-IRAN
e-mails: havangi@eetd.kntu.ac.ir, manekoui@eetd.kntu.ac.ir, teshnehlab@eetd.kntu.ac.ir

Received: 27.04.2010

Abstract

FastSLAM is a framework for simultaneous localization and mapping (SLAM) using a Rao-Blackwellized particle filter. However, FastSLAM degenerates over time. This degeneracy is due to the fact that a particle set estimating the pose of the robot loses its diversity. One of the main reasons for losing particle diversity in FastSLAM is sample impoverishment. In this case, most of the particle weights are insignificant. Another problem of FastSLAM relates to the design of an extended Kalman filter (EKF) for the landmark position's estimation. The performance of the EKF and the quality of the estimation depend heavily on correct a priori knowledge of the process and measurement noise covariance matrices (Q_t and R_t), which are, in most applications, unknown. Incorrect a priori knowledge of Q_t and R_t may seriously degrade the performance of the Kalman filter. This paper presents a modified FastSLAM framework by soft computing. In our proposed method, an adaptive neuro-fuzzy extended Kalman filter is used for landmark feature estimation. The free parameters of the adaptive neuro-fuzzy inference system (ANFIS) are trained using the steepest gradient descent (SD) to minimize the differences of the actual value of the covariance of the residual from its theoretical value as much possible. A novel multiswarm particle filter is then presented to overcome the impoverishment of FastSLAM. The multiswarm particle filter moves samples toward the region of the state space in which the likelihood is significant, without allowing them to go far from the region of significant proposal distribution. The simulation results show the effectiveness of the proposed algorithm.

Key Words: *SLAM, mobile robot, particle filter, particle swarm optimization, neuro-fuzzy, extended Kalman filter, gradient descent*

1. Introduction

Simultaneous localization and mapping (SLAM) is a fundamental problem of robots in performing autonomous tasks such as exploration in an unknown environment. It represents an important role in the autonomy of a mobile robot. The 2 key computational solutions to SLAM are the extended Kalman filter (EKF-SLAM) and the Rao-Blackwellized particle filter (FastSLAM). The EKF-SLAM approach is the oldest and the most popular approach to solving SLAM. To date, extensive research works have been reported applying EKF to the SLAM problem [1-3]. Several applications of EKF-SLAM have been developed for indoor applications [2,4],

*Corresponding author: Faculty of Electrical Engineering, K.N. Toosi University of Technology, Tehran-IRAN

outdoor applications [5], underwater applications [6], and underground applications [7]. However, EKF-SLAM suffers from 2 major problems: computational complexity and data association [8]. Recently, the FastSLAM algorithm approach has been proposed as an alternative approach to solve the SLAM problem. FastSLAM is an instance of the Rao-Blackwellized particle filter, which partitions the SLAM posterior into a localization problem and an independent landmark position estimation problem. In FastSLAM, a particle filter is used for the mobile robot position estimation and an EKF is used for the feature location's estimation. The key feature of FastSLAM, unlike EKF-SLAM, is the fact that 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. This characteristic gives FastSLAM the possibility of dealing with multihypothesis association problems. The ability to simultaneously pursue multiple data associations makes FastSLAM significantly more robust for data association problems than algorithms based on incremental maximum likelihood data association such as EKF-SLAM. The other advantage of FastSLAM over EKF-SLAM arises from the fact that particle filters can cope with nonlinear and non-Gaussian robot motion models, whereas EKF approaches approximate such models via linear functions. However, FastSLAM also has some drawbacks. In [9-13], it was noted that FastSLAM degenerates over time. This degeneracy is due to the fact that a particle set estimating the pose of the robot loses its diversity. One of the main reasons for losing particle diversity in FastSLAM is sample impoverishment. This occurs when likelihood lies in the tail of the proposal distribution [14]. On the other hand, FastSLAM highly relies on the number of particles to approximate the distribution density. Researchers tried to solve those problems in [14-22]. In all of these studies, the reliability of measurement plays a crucial role in the performance of the algorithm, where only additive noise is considered. In [23], to solve the sample impoverishment of the particle filter in mobile robot localization, a novel multiswarm particle filter was presented to estimate the posterior probability density of a robot's pose. In the current paper, we consider the impoverishment of a Rao-Blackwellized particle filter and propose a novel multiswarm Rao-Blackwellized particle filter to solve this problem. The novel multiswarm particle filter is proposed for estimating the posterior path of the robot. The multiswarm particle filter moves samples toward the region of the state space in which the likelihood is significant, without allowing them to go far from the region of significant proposal distribution. For this purpose, the multiswarm particle filter employs a conventional multiobjective optimization approach to weigh the likelihood and prior of the filter in order to alleviate the particle impoverishment problem. The minimization of the corresponding objective function is performed using the Gaussian particle swarm optimization (PSO) algorithm. Another problem of FastSLAM relates to the design of EKFs for landmark position estimation. A significant difficulty in designing an EKF can often be traced to incomplete a priori knowledge of the process covariance matrix Q_t and measurement noise covariance matrix R_t. In most applications, these matrices are unknown. Incorrect a priori knowledge of Q_t and R_t may seriously degrade the Kalman filter's performance [24,25]. Several research studies have been reported in the literature on adaptive Kalman filter algorithms [26-29]. In these works, the system under consideration is a simple one in which the size of the state vector is fixed. In addition, in [26], an adaptive neuro-fuzzy EKF for robot pose estimation was proposed. Here, only the parameters of the final stage of the adaptive neuro-fuzzy inference system (ANFIS) are trained using the steepest gradient descent (SD). Because the total number of states is fixed (tree state), the inputs to the ANFIS are also fixed. In this paper, as the Rao-Blackwellized particle filter is used for SLAM, N conditional landmark posteriors are estimated using an adaptive neuro-fuzzy EKF (N is the number of features that are time-varying). In the proposed method, the parameters of the antecedent part and the conclusion part of the ANFIS are trained using the SD.

2. The SLAM problem

The goal of SLAM is to simultaneously localize a robot and determine an accurate map of the environment. To describe SLAM, let us denote the map by Θ and the pose of the robot at time t by s_t. 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. In this situation, the SLAM problem can be formulized in a Bayesian probabilistic framework by representing each of the robot's positions and map locations as a probabilistic density function:

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

In essence, it is necessary to estimate the posterior density of maps Θ and poses s_t, given that we know the observation $z^t = \{z_1, ..., z_t\}$, the control input $u^t = \{u_1, ..., u_t\}$, and the data association n^t. Here, data association represents the mapping between the map points in Θ and observations in z^t. The SLAM problem is then achieved by applying Bayesian filtering, as follows.

$$p(s_t, \Theta | z^t, u^t, n^t) \propto p(z_t | s_t, \Theta, n_t) p(s_t, \Theta | z^{t-1}, u^t, n^t), \tag{2}$$

with:

$$p(s_t, \Theta | z^{t-1}, u^t, n^t) = \int p(s_t | s_{t-1}, u_t) p(s_{t-1}, \Theta | z^{t-1}, u^t, n^t) ds_{t-1}, \tag{3}$$

where $p(s_t | s_{t-1}, u_t)$ is the dynamics motion model and $p(z_t | s_t, \Theta, n^t)$ is the measurement model. The standard Bayesian solution for Eqs. (2) and (3) can be extremely expensive for high-dimensional maps Θ. However, Eq. (3) cannot be computed in closed form. FastSLAM is an efficient algorithm for the SLAM problem that is based on a straightforward factorization, as follows [12,13,30]:

$$p(s^t, \Theta | z^t, u^t, n^t) = p(s^t | z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n | s^t, z^t, u^t, n^t), \tag{4}$$

where $s^t = \{s_1, ..., s_t\}$ is a robot path or trajectory. This factorization states that the SLAM problem can be decomposed into estimation of the product of a posterior over the robot path and N landmark posteriors given the knowledge of the robot's path. The FastSLAM algorithm implements the path estimator $p(s^t | z^t, u^t, n^t)$ using a particle filter. The landmarks' poses $p(\theta_n | s^t, z^t, u^t, n^t)$ are realized by the EKF, using separate filters for different landmarks. Each particle forms the following [12,31]:

$$s_t^{[m]} =< s^{t,[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, ..., \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} >, \tag{5}$$

where $[m]$ indicates the index of the particle, $s^{t,[m]}$ is the mth particle's path estimate, and $\mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]}$ are the mean and the covariance of the Gaussian distribution representing the nth feature location conditioned on the path $s^{t,[m]}$. The updated algorithm of posteriors in FastSLAM can be described as follows.

1. Sampling a new pose
2. Updating the observed landmark
3. Calculating importance weight
4. Initializing feature
5. Resampling

In following subsections, we give details of the main steps.

2.1. Sampling a new pose

As mentioned in the previous section, FastSLAM employs a particle filter for estimating the path posterior $p(s^t|z^t, u^t, n^t)$ by a particle filter. The particle set S_t is calculated from the set S_{t-1} at time $t-1$, the control u_t, and the observation z_t. In general, it is not possible to draw samples directly from the SLAM posterior. Instead, the samples are drawn from a simpler distribution called the proposal distribution, $q(s^{t,[m]}|z^t, u^t, n^t)$. The mismatch between the SLAM posteriors and the actual proposal distribution values can be corrected using a technique called importance sampling. Therefore, in regions in which the target distribution is larger than the proposal distribution, the samples are assigned a larger weight, and in regions in which the target distribution is smaller than the proposal distribution, the samples will be given lower weights. For FastSLAM, the importance weights of each particle can be calculated as:

$$w_t^{[m]} = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(s^{t,[m]}|z^t, u^t, n^t)}{q(s^{t,[m]}|z^t, u^t, n^t)}. \tag{6}$$

As a result, the importance weight of each particle is equal to the ratio of the slam posterior and the proposal distribution. The proposal $q(s^{t,[m]}|z^t, u^t, n^t)$ can be represented by a recursive form, as:

$$\begin{aligned} q(s^{t,[m]}|z^t, u^t, n^t) &= q(s_t^{[m]}|s^{t-1,[m]}, z^t, u^t, n^t) q(s^{t-1,[m]}|z^t, u^t, n^t)^{Markov} \\ &= q(s_t^{[m]}|s^{t-1,[m]}, z^t, u^t, n^t) q(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1}). \end{aligned} \tag{7}$$

Similarly, the posterior can also be given by a recursive formula, using Bayes' theorem as follows:

$$p(s^{t,[m]}|z^t, u^t, n^t) = \eta p(z_t|s^{t,[m]}, z^{t-1}, u^t, n^t) p(s_t^{[m]}|z^{t-1}, u^t, n^t) p(s^{t-1,[m]}|z^{t-1}, u^{t-1}, n^{t-1}), \tag{8}$$

where η is a normalizing constant. With Eqs. (7) and (8), a sequential importance weight of the mth particle can be obtained as follows:

$$w_t^{[m]} = \eta w_{t-1}^{[m]} \frac{p(z_t|s^{t,[m]}, z^{t-1}, u^t, n^t) p(s_t^{[m]}|z^{t-1}, u^t, n^t)}{q(s_t^{[m]}|s^{t-1,[m]}, z^t, u^t, n^t)}. \tag{9}$$

The choice of the proposal distribution $q(s_t^{[m]}|s^{t-1,[m]}, z^t, u^t, n^t)$ is one of the most critical issues in the design of a FastSLAM. Two of those critical issues are as follows: samples are drawn from the proposal distribution, and the proposal distribution is used to evaluate the importance weights. The optimal importance density function that minimizes the variance of the importance weights is the following equation (FastSLAM 2.0 [13]):

$$q(s^{t,[m]}|z^{t-1}, u^t, n^{t-1}) = p(s^{t,[m]}|z^{t-1}, u^t, n^{t-1}). \tag{10}$$

However, there are some special cases in which the use of the optimal importance density is possible. The main disadvantage of these approaches is that they are more difficult to implement. The most popular suboptimal choice is the transitional prior (FastSLAM 1.0 [32]):

$$q(s^{t,[m]}|z^{t-1}, u^t, n^{t-1}) = p(s_t|u_t, s_{t-1}^{[m]}). \tag{11}$$

In this paper, FastSLAM 1.0 was used due to its easy calculation. Hence, by substitution of Eq. (11) into Eq. (9), the weight's updated equation is:

$$w_t^{[m]} = \eta w_{t-1}^{[m]} p(z_t|s^{t,[m]}, z^{t-1}, u^t, n^t). \tag{12}$$

Note that the $w_k^{[m]}$ values are updated incrementally while considering a full trajectory of the robot state $s^{t,[m]}$.

2.2. Updating the observed landmark

FastSLAM represents the posterior landmark estimates$p(\theta_n|s^t, z^t, u^t, n^t)$ using low-dimensional EKFs. In fact, FastSLAM updates the posterior over the landmark estimates, respected by the mean $\mu_{n,t-1}^{[m]}$ and the covariance $\Sigma_{n,t-1}^{[m]}$. The updated values $\mu_{n,t}^{[m]}$ and $\Sigma_{n,t}^{[m]}$ are then added to the temporary particle set S_t, along with the new pose. The update depends on whether or not a landmark n was observed at time t. For $n \neq n_t$, the posterior over the landmark remains unchanged, as follows [31].

$$\mu_{n,t}^{[m]} = \mu_{n,t-1}^{[m]} \tag{13}$$

$$\Sigma_{n,t}^{[m]} = \Sigma_{n,t-1}^{[m]} \tag{14}$$

For the observed feature $n = n_t$, the update is specified through the following equation [31].

$$\begin{aligned} p(\theta_{n_t}|s^{t,[m]}, n^t, z^t) &= \frac{p(z_t|\theta_{n_t}, s^{t,[m]}, n^t, z^{t-1}) p(\theta_{n_t}|s^{t,[m]}, n^t, z^{t-1})}{p(z_t|s^{t,[m]}, n^t, z^{t-1})} \\ &= \eta \underbrace{p(z_t|\theta_{n_t}, s^{t,[m]}, n^t, z^{t-1})}_{\sim N(z_t, g(\theta_{n_t}, s_t^{[m]}, R_t)} \underbrace{p(\theta_{n_t}|s^{t,[m]}, n^t, z^{t-1})}_{\sim N(\theta_{n_t}, \mu_{n,t-1}^{[m]}, \Sigma_{n,t-1}^{[m]})} \end{aligned} \tag{15}$$

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 $\mu_{n,t-1}^{[m]}$ and covariance $\Sigma_{n,t-1}^{[m]}$. For the new estimate at time t to also be Gaussian, FastSLAM linearizes the perceptual model $p(z_t|\theta_{n_t}, s^{t,[m]}, n^t, z^{t-1})$ by an EKF. In particular, FastSLAM approximates the measurement function g by the following first-degree Taylor expansion [31]:

$$g(\theta_{n_t}, s_t^{[m]}) = \underbrace{g(\mu_{n,t-1}^{[m]}, s_t^{[m]})}_{\hat{z}_t^{[m]}} + \underbrace{g'(s_t^{[m]}, \mu_{n,t-1}^{[m]})}_{G_t^{[m]}}(\theta_{n_t} - \mu_{n,t-1}^{[m]}) = \hat{z}_t^{[m]} + G_t^{[m]}(\theta_{n_t} - \mu_{n,t-1}^{[m]}). \tag{16}$$

Under this approximation, the posterior of landmark n_t is indeed Gaussian. The mean and covariance are obtained using the following measurement updates.

$$\hat{z}_t = g(s_t^{[m]}, \mu_{n_t,t-1}) \tag{17}$$

$$G_{\theta_{n_t}} = \nabla_{\theta_{n_t}} g(s_t, \theta_{n_t})|_{s_t = s_t^{[m]}; \theta_{n_t} = \mu_{n_t,t-1}^{[m]}} \tag{18}$$

$$Z_{n,t} = G_{\theta_{n_t}} \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T + R_t \tag{19}$$

$$K_t = \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T Z_{n,t}^{-1} \tag{20}$$

$$\mu_{n_t,t}^{[m]} = \mu_{n_t,t-1}^{[m]} + K_t(z_t - \hat{z}_t) \tag{21}$$

$$\Sigma_{n_t,t}^{[m]} = (I - K_t G_{\theta_{n_t}}) \Sigma_{n_t,t-1}^{[m]} \tag{22}$$

2.3. Calculating importance weight

In FastSLAM 1.0, the importance weight should be computed by considering the most recent observation, and it is given by Eq. (9) as the following:

$$w_t^{[m]} = w_{t-1}^{[m]} \frac{p(z_t|s^{t,[m]}, z^{t-1}, u^t, n^t)p(s_t^{[m]}|u^t, n^t)}{p(s_t^{[m]}|u^t, n^t)} = w_{t-1}^{[m]} p(z_t|s^{t,[m]}, z^{t-1}, u^t, n^t). \tag{23}$$

2.4. Initializing feature

A new feature is initialized as a function of the robot pose $s_t^{[m]}$ and measurement z_t. The feature mean $\mu_{n.t}^{[m]}$ and the feature covariance $\Sigma_{n,t}^{[m]}$ in the feature initialization are calculated as follows [12,13].

$$\mu_{n.t}^{[m]} = g^{-1}(z_t, s_t^{[m]}) \tag{24}$$

$$\Sigma_{n,t}^{[m]} = (G_{\widehat{n}}^{[m]} R_t^{-1} G_{\widehat{n}}^{[m]T})^{-1} \tag{25}$$

2.5. Resampling

Since the variation of the importance weights increases over time, resampling plays a vital role in FastSLAM [33,34]. In the resampling process, particles with low importance weight are eliminated and particles with high weights are multiplied. After the resampling, all particle weights are then reset to:

$$w_t^{[m]} = \frac{1}{N}. \tag{26}$$

This enables FastSLAM to estimate increasing environmental states defiantly without growing a number of particles. However, resampling can delete good samples from the sample set, and, in the worst case, the filter diverges. The decision on how to determine when to resample is a fundamental issue. Liu introduced the so-called effective number of particles, N_{eff}, to estimate how well the current particle set represents the true posterior [18]. This quality is computed as:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} w_t^{[i]}}, \tag{27}$$

where w_t^i refers to the normalized weight of particle i. The resampling process is operated whenever N_{eff} is below a predefined threshold, N_{tf}. Here, N_{tf} is usually a constant value, such as:

$$N_{tf} = \frac{3}{4}M, \tag{28}$$

where M is the number of particles.

3. A modified FastSLAM framework using soft computing

In this section, a modified FastSLAM framework using soft computing is presented in detail. In modified FastSLAM, a multiswarm particle filter is used to overcome the impoverishment of the particle filter and an adaptive neuro-fuzzy EKF is proposed for features estimation.

3.1. A modified sampling of FastSLAM

FastSLAM relies on importance sampling; in other words, it uses proposal distributions to approximate the posterior distribution. The most common choice of proposal distribution, also used in this paper, is the probabilistic model of the states' evolution, or the transition prior $p(s_t|u_t, s_{t-1}^{[m]})$. Because the proposal distribution is suboptimal, there are 2 serious problems in the particle filter. One problem is sample impoverishment, which occurs when the likelihood $p(z_t|s^{t,[m]}, z^{t-1}, u^t, n^t)$ is very narrow or the likelihood lies in the tail of the proposal distribution. The prior distribution is effective when the observation accuracy is low. It is not effective, however, when the prior distribution is a much broader distribution than the likelihood. Hence, in the updating step, only a few particles will have significant importance weights. This problem implies that a large computational effort is devoted to updating the particles with negligible weight. Thus, the sample set only contains a few dissimilar particles, and sometimes they will drop to a single sample after several iterations. As a result, important samples may be lost. The second problem of the particle filter is the number of particles' dependency that estimates the pose of the robot. If the number of particles is small, then there might not have been particles distributed around the true pose of the robot. After several iterations, it is then very difficult for particles to converge to the true pose of the robot. For standard FastSLAM, there is one method to solve the problem. This is to augment the number of the particles. However, this would make the computational complexity unacceptable. To solve these problems in FastSLAM, PSO is considered to optimize the sampling process of FastSLAM.

3.1.1. Particle swarm optimization

PSO is a population-based search algorithm based on the simulation of the social behavior of birds in a flock. PSO is initialized with a group of random particles and then computes the fitness of each one. Finally, it can find the best solution in the problem space via many repeating iterations. In each iteration, each particle keeps track of its coordinates that are associated with the best solution it has achieved so far (pbest) and the coordinates that are associated with the best solution achieved by any particle in its neighborhood (gbest). Supposing that the search space dimension is D, and the number of particles is N, the position and velocity of the ith particle are represented by $x_i = (x_{i1}, ..., x_{iD})$ and $v_i = (v_{i1}, ..., v_{iD})$, respectively. Let $P_{bi} = [p_{i1}, ...p_{iD}]$ denote the best position that particle i has achieved so far, and P_g the best of P_{bi} for any $i = 1, ..., N$. The PSO algorithm could be performed by the following equations.

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{29}$$

$$\vec{v}_i(t) = w\vec{v}_i(t-1) + c_1 r_1(\vec{P}_{bi} - \vec{x}_i(t-1)) + c_2 r_2(\vec{P}_g - \vec{x}_i(t-1)) \tag{30}$$

Here, $t$represents the iteration number and c_1, c_2 are the learning factors. Usually, $c_1 = c_2 = 2$, and r_1, r_2 are random numbers in the interval $(0, 1)$, while w is the inertial factor. Note that the larger the value of w, the wider the search range.

3.1.2. A multiswarm particle filter

As discussed in the previous section, impoverishment occurs when the number of particles in the high-likelihood area is low. We addressed this problem by intervening in localization based on a particle filter after the generation of the samples in the prediction phase and before resampling. The aim is to move these samples toward a region

of the state space in which the likelihood is significant, without allowing them to go far from the region of the significant prior. For this purpose, we consider a multiobjective function, as follows [23]:

$$F = F_1 + F_2. \tag{31}$$

The first objective, F_1, consists of a function that is maximized at regions of high likelihood, as follows.

$$F_1 = e^{-\frac{1}{2}(z_t - \hat{z}_{n_t,t})^T \left[Z_{n_t,t}\right]^{-1} (z_t - \hat{z}_{n_t,t})} \tag{32}$$

Here, $Z_{n_t,t}$ is the residual covariance matrix defined in Eq. (19), $\hat{z}_{n_t,t}$ is the predicted measurement, and z_t is the actual measurement. The second objective, F_2, is maximized at regions of high prior.

$$F_2 = e^{-\frac{1}{2}(x_t - \hat{x}_t)^T [Q_t]^{-1} (x_t - \hat{x}_t)}, \tag{33}$$

where Q_t is the measurement noise covariance matrix. We use a simple idea to solve this problem. The basic idea is that particles are encouraged to be in the region of high likelihood by incorporating the current observation without allowing them to go far from the region of significant prior before the sampling process. This implies that a simple and effective method for this purpose is the use of PSO. In fact, by using PSO, we can move all of the particles toward the region that maximizes the objective function F before the sampling process. For this purpose, we consider a fitness function, as follows:

$$\text{Fitness(t)} = \frac{1}{2}(z_t - \hat{z}_{n_t,t})^T \left[Z_{n_t,t}\right]^{-1} (z_t - \hat{z}_{n_t,t}) + \frac{1}{2}(x_t - \hat{x}_t^-) Q_t^{-1} (x_t - \hat{x}_t^-)^T. \tag{34}$$

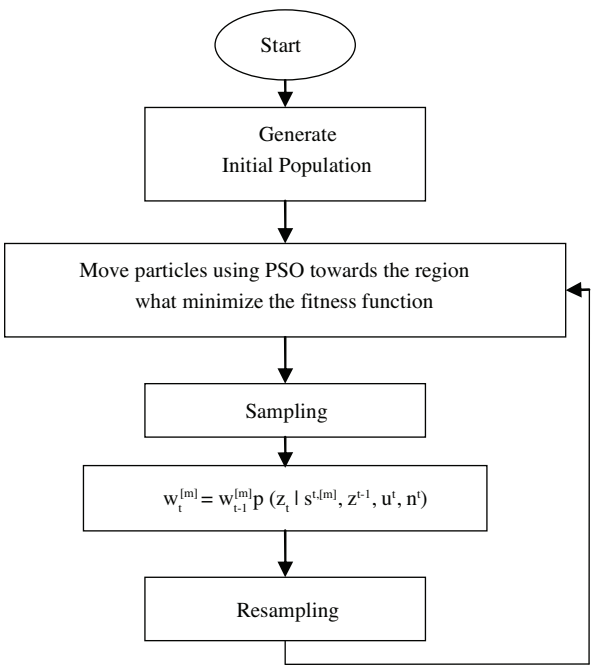

Figure 1. Multiswarm particle filter.

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

specified before using it. Most approaches use uniform probability distribution to generate random numbers. However, it is difficult to obtain fine tuning of the solution and escape from local minima using a uniform distribution. Hence, we use velocity updates based on the Gaussian distribution. In this situation, there is no more need to specify the parameter learning factors c_1 and c_2. Furthermore, using the Gaussian PSO, the inertial factor ω is set to zero and an upper bound for the maximum velocity $v_{\max}$ is not necessary anymore [35]. Thus, the only parameter to be specified by the user is the number of particles. Initial values of the particle filter are selected as the initial population of the PSO. Initial velocities of the PSO are set equal to zero. The PSO algorithm updates the velocity and position of each particle by the following equations [35].

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{35}$$

$$\vec{v}_i(t) = |randn| \left(P_{pbest} - \vec{x}_i(t-1)\right) + |randn| \left(P_{gbest} - \vec{x}_i(t-1)\right) \tag{36}$$

The PSO moves all particles toward the particle with best fitness. When the best fitness value reaches a certain threshold, the optimized sampling process is stopped. With this set of particles, the sampling process will be done on the basis of proposal distribution. The corresponding weights will be as follows:

$$w_k^{[m]} = w_{k-1}^{[m]} p(z_t | s^{t,[m]}, z^{t-1}, u^t, n^t), \tag{37}$$

where:

$$p(z_t | s^{t,[m]}, z^{t-1}, u^t, n^t) = \frac{1}{\sqrt{(2\pi)|Z_{n_t,t}|}} \exp\{-\frac{1}{2}(z_t - \hat{z}_{n_t,t})^T \left[Z_{n_t,t}\right]^{-1} (z_t - \hat{z}_{n_t,t})\}. \tag{38}$$

The flowchart for the proposed algorithm is shown in Figure 1. As observed, this algorithm can be described with the following steps.

Step 1. General initial population initialization

1. Initialize particle velocity
2. Initialize particle position
3. Initialize particle fitness value
4. Initialize pbest and gbest

Step 2. **Move particles using PSO toward the region that minimizes the fitness function**

Adjust the speed and location of particles.

Step 3. Important sampling

1. Update the particle state
   a) Time update
   b) Measurement update
2. Calculate the mean value $s_t^{[m]}$ and covariance $P_t^{[m]}$ of particle set $\{\hat{x}_k^i\}_{i=1}^N$
3. Draw sample from proposal distribution

Step 4. Weight updates

1. Calculate the weight

2. Weight will be normalized

$$\bar{w}_t^{[i]} = \frac{w_t^{[i]}}{\sum_{i=1}^{N} w_t^{[i]}} \tag{39}$$

Step 5. Resampling

The resampling is operated whenever N_{eff} is below a predefined threshold.

Step 6. Prediction

Compute the posterior probability estimates.

$$\hat{x}_t = \sum_{i=1}^{N} s_t^{[i]} \bar{w}_t^{[i]} \tag{40}$$

Step 7. Compute the global optimum p^g at current time

Set the particle that has the highest weight value of current particles as p_k^g.

Step 8. Increase time k, return to step 2 to recursively estimate the posterior probability

3.2. Feature update by adaptive neuro-fuzzy EKF

As stated earlier, FastSLAM represents the conditional landmark estimates $p(\theta_n|s^t, z^t, u^t, n^t)$ by EKF. The traditional EKF assumes complete a priori knowledge of the process and measurement noise statistics, matrices Q_t and R_t. However, in most applications, these matrices are unknown. An incorrect a priori knowledge of Q_t and R_tm ay lead to performance degradation [25] and can even lead to practical divergence [24,25]. One of the efficient ways to overcome the above weakness is to use an adaptive algorithm. Two major approaches that have been proposed for adaptive EKFs are multiple-model adaptive estimation (MMAE) and innovation-based adaptive estimation (IAE) [24,25]. Although the implementation of these approaches is different, they both share the same concept of utilizing new statistical information obtained from the residual (innovation) sequence. As landmarks are static, we assume that the noise covariance Q_t is completely known. Hence, the algorithm to estimate the measurement noise covariance R_t can be derived. The adaptation is adaptively adjusting the measurement noise covariance matrix R_t by using a neuro-fuzzy system. In this case, an IAE algorithm to adapt the measurement noise covariance matrix R_t is derived. The technique known as covariance matching is used. The basic idea behind this technique is to make the actual value of the covariance of the residual be consistent with its theoretical value. The innovation sequence $r_t = (z_t - \hat{z}_{n_t,t})$ has a theoretical covariance that is obtained from the EKF algorithm.

$$S_t = G_{\theta_{n_t}} \Sigma_{n_t,t-1}^{[m]} G_{\theta_{n_t}}^T + R_t \tag{41}$$

The actual residual covariance $\hat{C}_k$ can be approximated by its sample covariance, through averaging inside a moving window of size N as follows:

$$\hat{C}_t = \frac{1}{N} \sum_{i=k-N+1}^{t} (r_i^T r_i). \tag{42}$$

If the actual value of covariance $\hat{C}_t$ has discrepancies with its theoretical value, then the diagonal elements of R_t based on the size of this discrepancy can be adjusted. The objective of these adjustments is to correct this mismatch as far as possible. The size of the mentioned discrepancy is given by a variable called the degree of mismatch (DOM_t), defined as:

$$DOM_t = S_t - \hat{C}_t. \tag{43}$$

The basic idea used to adapt matrix R_t is as follows: from Eq. (41), an increment in R_t will increase S_t and vice versa. 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. The general rules of adaptation are as follows.

If $DOM_t(i,i) \cong 0$, then maintain R_t unchanged.

If $DOM_t(i,i) > 0$, then decrease R_t.

If $DOM_t(i,i) < 0$, then increase R_t.

In this paper, the IAE adaptive scheme of the EKF coupled with an ANFIS is presented to adjust R_t. As the size of DOM_t and R_t is 2, a 2-system ANFIS is used to adjust the EKF. The structure of one of these systems is described in the next section.

3.2.1. The ANFIS architecture

The ANFIS model has been considered as a 2-input, single-output system. The inputs of the ANFIS are DOM_t and $DeltaDOM_t$. Here, $DeltaDOM_t$ is defined as [26]:

$$DeltaDOM_t = DOM_t - DOM_{t-1}. \tag{44}$$

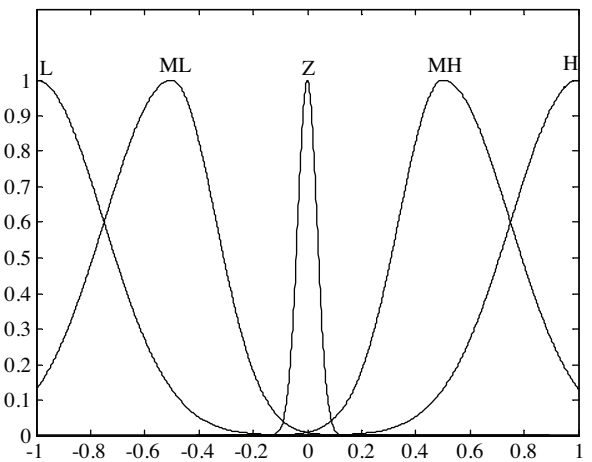

Figure 2. Membership functions $DOM_t(i,i)$ and $DeltaDOM_t$.

Figure 2 presents membership functions for $DOM_t(i,i)$ and $DeltaDOM_t$. Finally, adjustment of R_t is performed using the following relation:

$$R_t = R_t + \Delta R_t, \tag{45}$$

where ΔR_t is the ANFIS output and the membership function of ΔR_t is shown in Figure 3. This ANFIS is a 5-layer network, as shown in Figure 4.

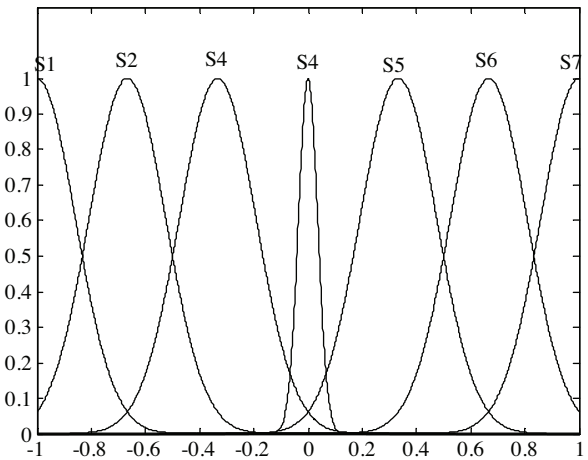

Figure 3. Membership function R_t.

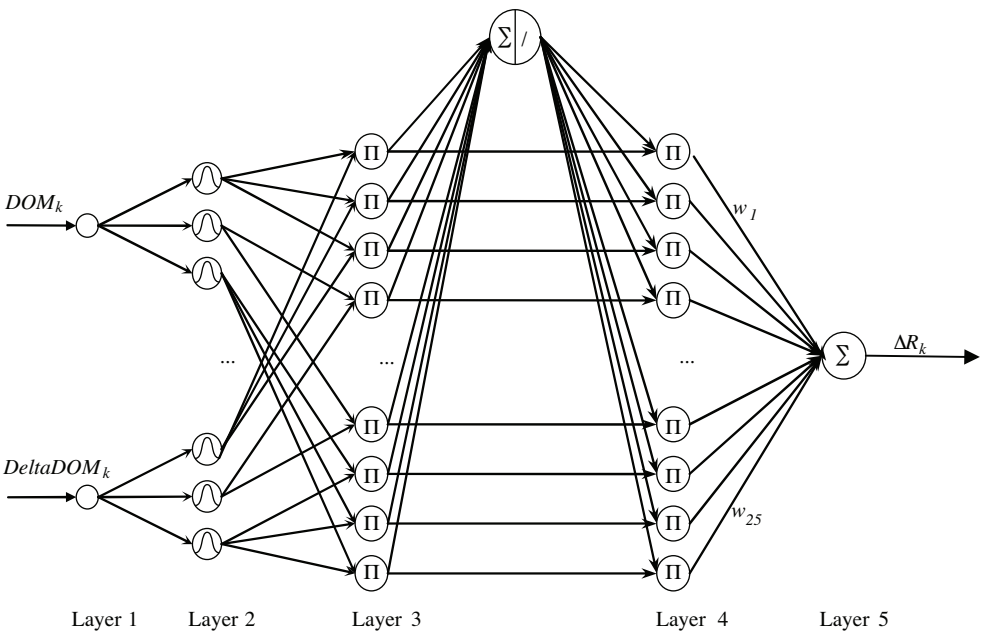

Figure 4. The neuro-fuzzy system for feature update.

Table. Rule table.

		$DeltaDoM_k$				
		L	LM	Z	HM	H
DoM_k	L	S7	S7	S6	S5	S4
	LM	S7	S6	S5	S4	S3
	Z	S6	S5	S4	S3	S2
	HM	S5	S4	S3	S2	S1
	H	S4	S3	S2	S1	S1

Let u_i^l and o_i^l denote the input to output from the ith node of the lth layer, respectively. To provide a clear understanding of an ANFIS, the functioning of layers 1 through 5 are elaborated below.

Layer 1: The node in this layer only transmits input values to the next layer directly, or:

$$o_i^1 = u_i^1. \tag{46}$$

Layer 2: In this layer, each node only performs a membership function. Here, the input variable is fuzzified, employing 5 membership functions (MFs). The output of the ith MF is given as:

$$o_{ij}^2 = \mu_{ij}(u^2) = \exp\left\{-\frac{(u_{ij}^2 - m_{ij})^2}{(\delta_{ij})^2}\right\}, \tag{47}$$

where m_{ij} and δ_{ij} are the mean and width of the Gaussian membership function, respectively. The subscript ij indicates the jth term of the ith input. Each node in this layer has 2 adjustable parameters: m_{ij} and δ_{ij}.

Layer 3: The nodes in this layer are rule nodes. A rule node performs a fuzzy AND operation (or product inference) for calculating the firing strength.

$$o_l^3 = \prod_i u_i^3 \tag{48}$$

Layer 4: The node in this layer performs the normalization of firing strengths from layer 3:

$$o_l^4 = \frac{u_l^4}{\sum_{l=1}^{9} u_l^4}. \tag{49}$$

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 of the normalization firing strengths from layer 4 with the corresponding singleton constituents and acts as a defuzzfier:

$$\Delta R_i = \sum_{l=1}^{25} u_l^5 w_l. \tag{50}$$

The fuzzy rules that complete the ANFIS rule base are shown in the Table.

3.2.2. Learning algorithm

The aim of the training algorithm is to adjust the network weights through the minimization of the following cast function:

$$E = \frac{1}{2}e_t^2, \tag{51}$$

where:

$$e_t = S_t - \hat{C}_t. \tag{52}$$

By using the backpropagation (BP) learning algorithm, the weighting vector of the ANFIS is adjusted such that the error defined in Eq. (41) is less than a desired threshold value after a given number of training cycles. The well-known BP algorithm may be written as:

$$W(t+1) = W(t) - \eta\frac{\partial E(t)}{\partial W(t)}. \tag{53}$$

Here, η and W represent the learning rate and tuning parameter of the ANFIS, respectively. Let $W = [m, \sigma, \delta]^T$ be the weighting vector of the ANFIS. The gradient of E with respect to an arbitrary weighting vector W is as follows:

$$\frac{\partial E(t)}{\partial W(t)} = -e_t\left(\frac{\partial \Delta R(t)}{\partial W(t)}\right). \tag{54}$$

By the recursive application of the chain rule, the error term for each layer is first calculated, and then the parameters in the corresponding layers are adjusted as follows.

$$m(t+1) = m(t) - \eta\frac{\partial E(t)}{\partial m(t)} \tag{55}$$

$$\sigma(t+1) = \sigma(t) - \eta\frac{\partial E(t)}{\partial \sigma(t)} \tag{56}$$

$$\delta(t+1) = \delta(t) - \eta\frac{\partial E(t)}{\partial \delta(t)} \tag{57}$$

4. Implementation and results

Simulation experiments were carried out to evaluate the performance of the proposed approach in comparison with the classical method. The proposed solution for the SLAM problem was tested for the benchmark environment, with varied numbers and positions of landmarks, in [36].

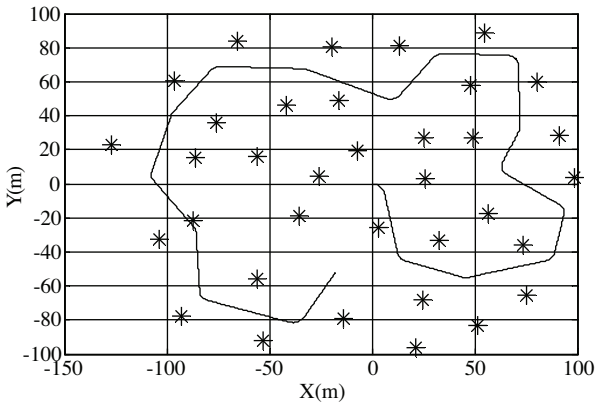

Figure 5. The experiment environment: the asterisks denote the landmark positions and the blue line is the path of the robot.

Figure 5 shows the robot's trajectory and landmark locations. The asterisks depict the locations of the landmarks that are known and stationary in the environment. The state of the robot can be modeled as (x, y, ϕ), such that (x, y) are the Cartesian coordinates and ϕ is the orientation, respectively, to the global environment. The kinematics equations for the mobile robot are in the following form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} (V + v_v)\cos(\phi + [\gamma + v_\gamma]) \\ (V + v_v)\sin(\phi + [\gamma + v_\gamma]) \\ \frac{(V+v_v)}{B}\sin(\gamma + v_\gamma) \end{bmatrix}, \tag{58}$$

where B is the base line of the vehicle and $u = \begin{bmatrix} V & \gamma \end{bmatrix}^T$ is the control input at time t consisting of a velocity input V and a steer input γ. The process noise, $v = \begin{bmatrix} v_v & v_\gamma \end{bmatrix}^T$, is assumed to be Gaussian. The vehicle is assumed to be equipped with a range-bearing sensor that provides a measurement of range r_i and bearing θ_i to an observed feature ρ_i relative to the vehicle. The observation z of feature ρ_i on the map can be expressed as:

$$\begin{bmatrix} r_i \\ \theta_i \end{bmatrix} = \begin{bmatrix} \sqrt{(x - x_i)^2 + (y - y_i)^2} + \omega_r \\ \tan^{-1}\frac{y - y_i}{x - x_i} - \phi + \omega_\theta \end{bmatrix}, \tag{59}$$

where (x_i, y_i) is the landmark position on the map, and $W = \begin{bmatrix} \omega_r & \omega_\theta \end{bmatrix}^T$ relates to observation noise. The initial position of the robot is assumed to be $x_0 = 0$. The robot moves at a speed of 3 m/s 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 frontal field of view of $180°$. The control noise is $\sigma_v = 0.3m/s$ and $\sigma_\gamma = 3°$. The control frequency is $40HZ$ and observation scans are obtained at $5HZ$. The measurement noise is $0.2m$ in range and $1°$ in bearing. Data association is assumed to be known. At first, the performance of the 2 algorithms can be compared by keeping the noise levels constant (process noise and measurement noise) and varying the number of particles. Figures 6-9 show the performance of the 2 algorithms for numbers of particles equivalent to 10 and 5. As observed, the modified FastSLAM is more accurate than the FastSLAM. Furthermore, the performance of the modified FastSLAM does not depend on the number of particles, while the performance of FastSLAM highly depends on the number of particles. For very low numbers of particles, FastSLAM diverges while the modified FastSLAM is completely robust. This is because the PSO in the modified FastSLAM places

the particles in the high likelihood region. In addition, we observed that the modified FastSLAM requires fewer particles than FastSLAM in order to achieve a given level of accuracy for state estimates.

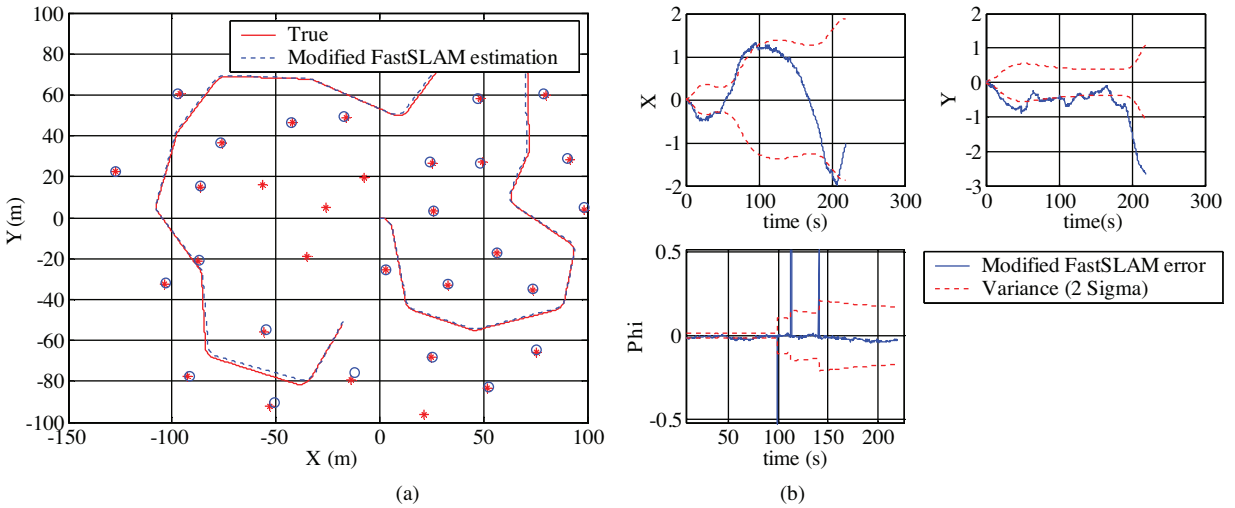

Figure 6. Modified FastSLAM: In this experiment, the control noise was $\sigma_v = 0.3m/s, \sigma_\gamma = 3^\circ$; the measurement noise was $\sigma_r = 0.2m$, $\sigma_\theta = 1^\circ$; and the number of particles was 10 (n = 10). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

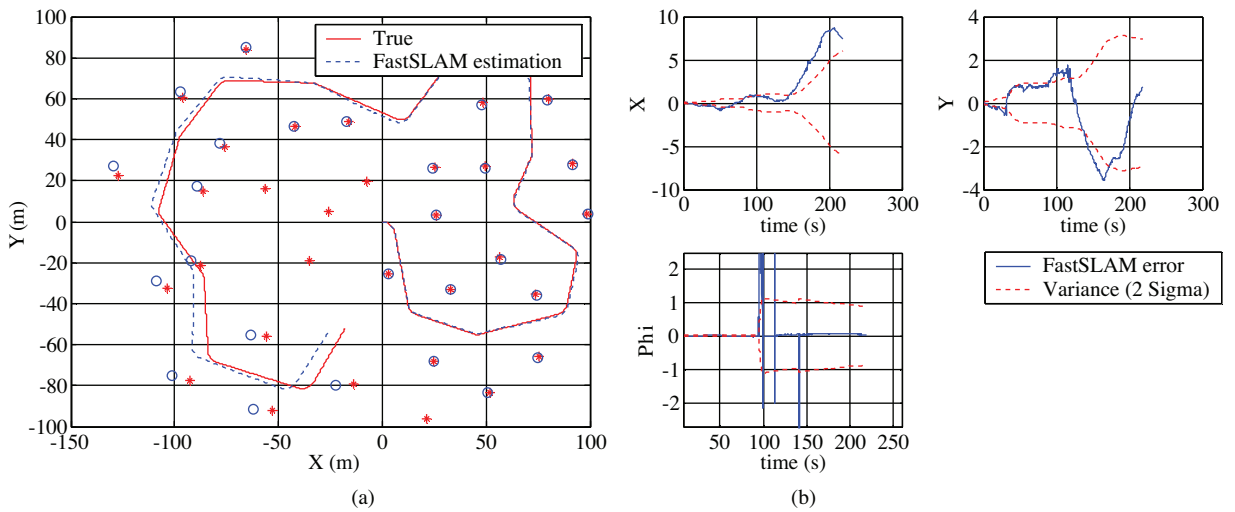

Figure 7. FastSLAM: In this experiment, the control noise was $\sigma_v = 0.3m/s, \sigma_\gamma = 3^\circ$; the measurement noise was $\sigma_r = 0.2m$, $\sigma_\theta = 1^\circ$; and the number of particles was 10 (n = 10). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

Next, we compared the performance of the 2 algorithms while varying the level of measurement noise as $\sigma_r = 0.01$ m, $\sigma_\theta = 0.1$ and fixing the control noise at $\sigma_v = 0.3m/s$, $\sigma_\gamma = 3^\circ$. All experiments were run with 5 particles in this case. From Figures 10 and 11, we can observe that the performance of our proposed

method will outperform the classical FastSLAM. This is because sample impoverishment occurs in FastSLAM. In fact, because the proposal distribution is broader than the likelihood in FastSLAM, the weight of most of the participles are insignificant when their likelihoods are evaluated. As a result, the estimation accuracy of FastSLAM will decrease whenever the robot's motion is noisier and the range-bearing sensor is more accurate. In this situation, the performance of FastSLAM is mostly reduced while the performance of the modified FastSLAM is almost fixed. For very low values of measurement errors, FastSLAM clearly begins to diverge while the proposed algorithm is robust.

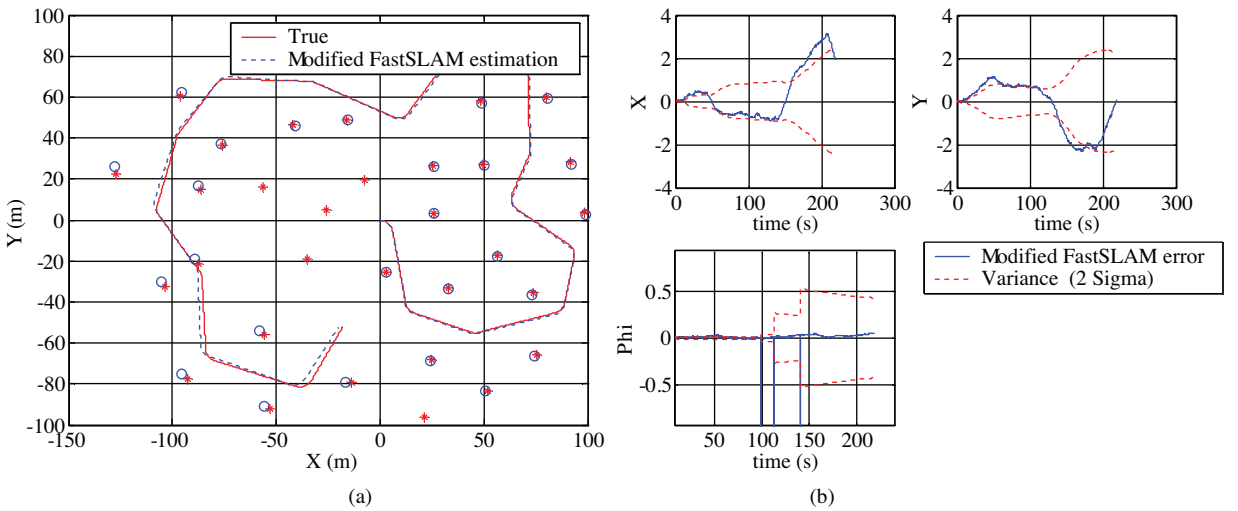

Figure 8. Modified FastSLAM: In this experiment, the control noise was $\sigma_v = 0.3m/s, \sigma_\gamma = 3^\circ$; the measurement noise was $\sigma_r = 0.2m$, $\sigma_\theta = 1^\circ$; and the number of particles was 5 (n = 5). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

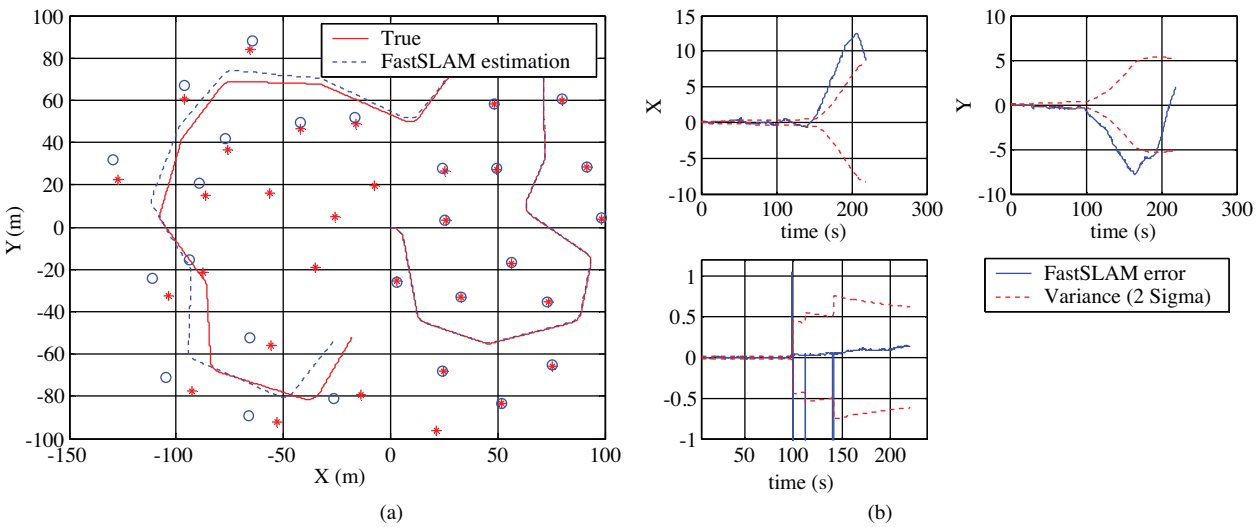

Figure 9. FastSLAM: In this experiment, the control noise was $\sigma_v = 0.3m/s, \sigma_\gamma = 3^\circ$; the measurement noise was $\sigma_r = 0.2m$, $\sigma_\theta = 1^\circ$; and the number of particles was 5 (n = 5). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

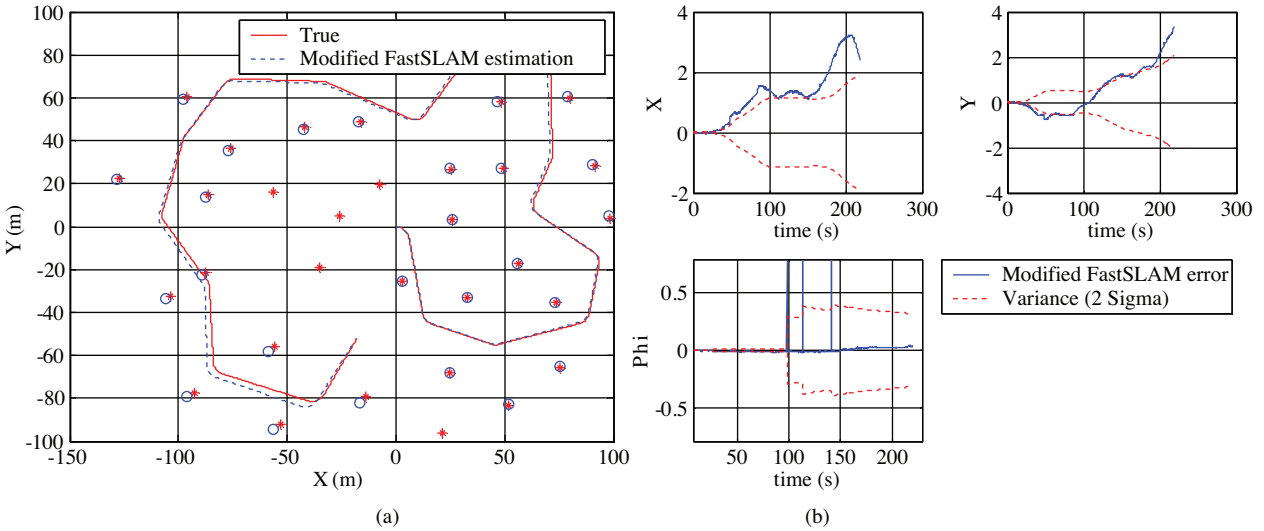

Figure 10. Modified FastSLAM: In this experiment, the control noise was $\sigma_v = 0.3m/s, \sigma_\gamma = 3^\circ$; the measurement noise was $\sigma_r = 0.01m$, $\sigma_\theta = 0.1^\circ$; and the number of particles was 5 (n = 5). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

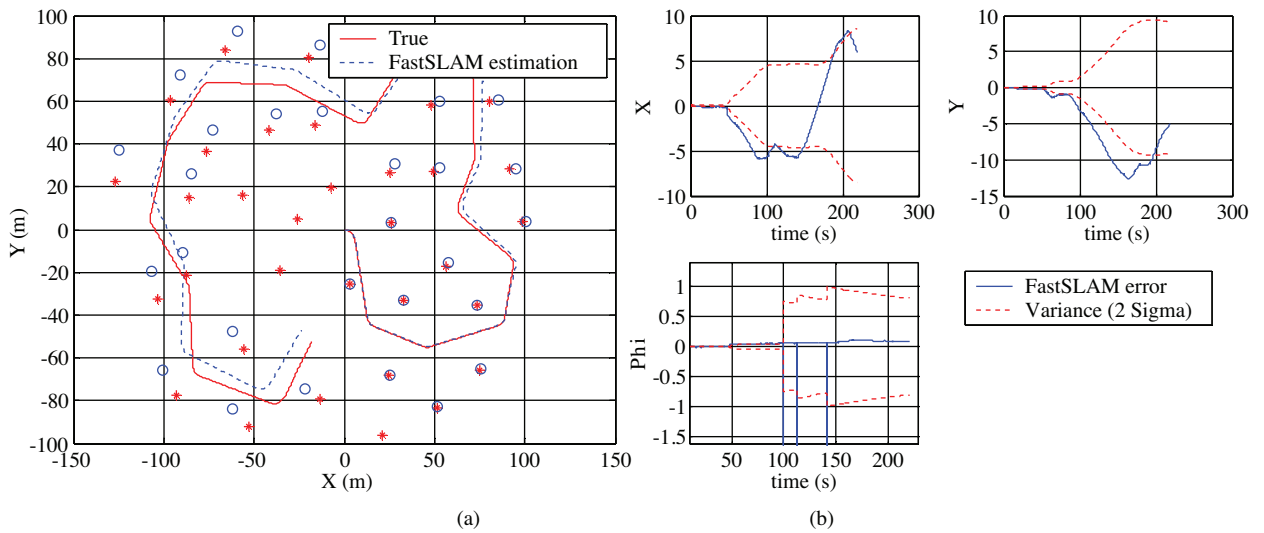

Figure 11. FastSLAM: In this experiment, the control noise was $\sigma_v = 0.3m/s, \sigma_\gamma = 3^\circ$; the measurement noise was $\sigma_r = 0.01m$, $\sigma_\theta = 0.1^\circ$; and the number of particles was 5 (n = 5). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

Finally, we studied the effect of the designed ANFIS on the performance of FastSLAM. For this purpose, we considered a situation in which the measurement noise was wrongly considered as $\sigma_r = 0.5$, $\sigma_\theta = 3.0$ and the control noise was rightly considered as $\sigma_v = 0.3m/s$, $\sigma_\gamma = 3^\circ$. The performance of the modified FastSLAM was compared with classical FastSLAM where its measurement covariance matrix R_t was kept static throughout the experiment. The proposed algorithm starts with wrongly known statistics, then adapts the R_t matrix through

the 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 the SD during training. Figures 12 and 13 show the comparison between the proposed algorithm and FastSLAM when the particle number is 20. It can

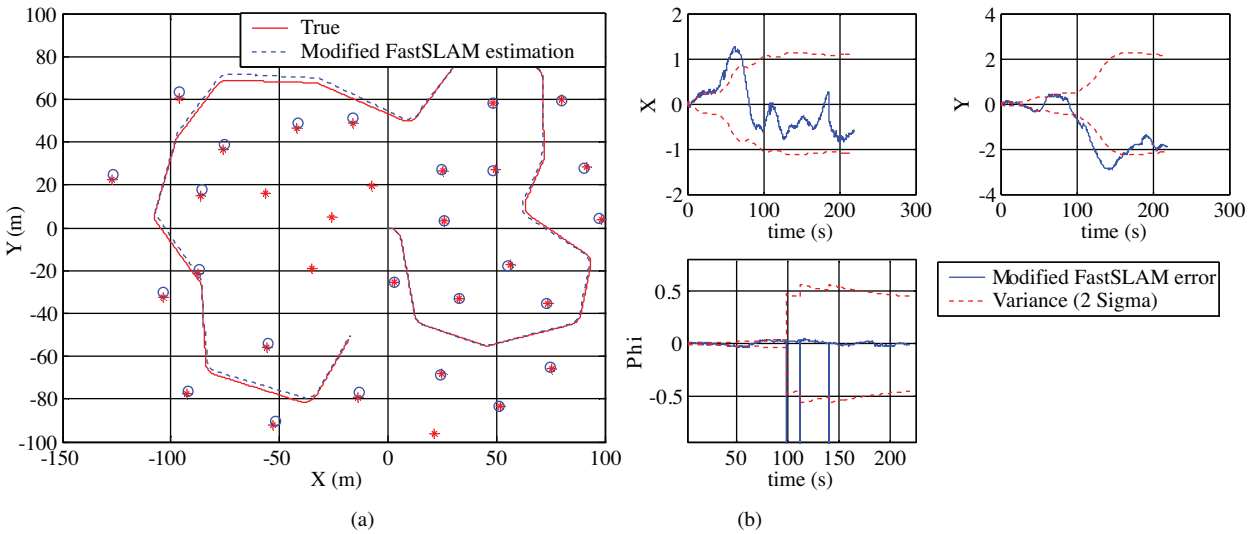

Figure 12. Modified FastSLAM: In this experiment, measurement noise was wrongly considered as $\sigma_r = 0.5$, $\sigma_\theta = 3.0$ and control noise was rightly considered as $\sigma_v = 0.3m/s$, $\sigma_\gamma = 3^\circ$. The number of particles was 5 (n = 5). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

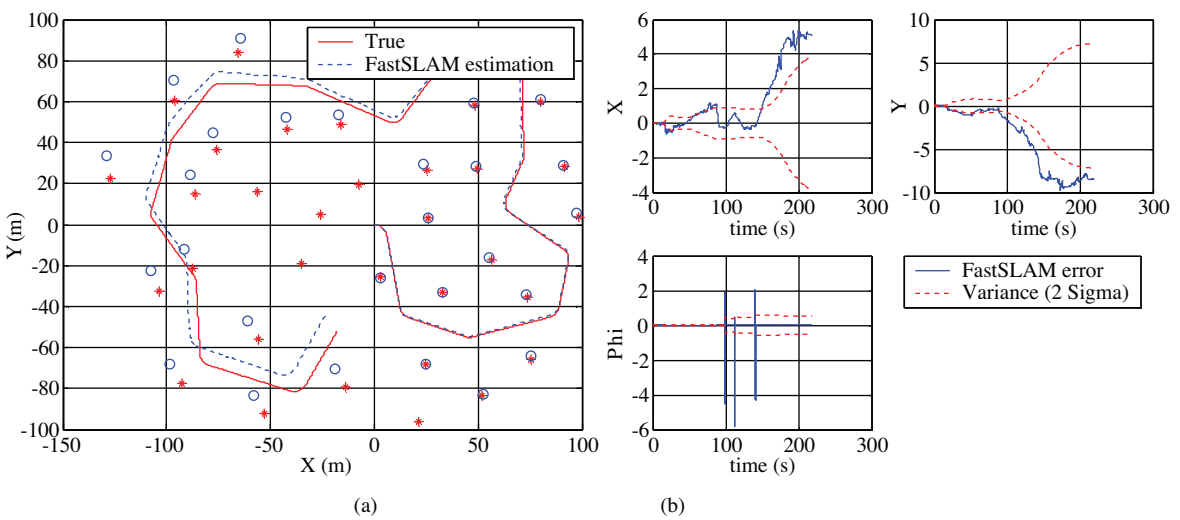

Figure 13. FastSLAM: In this experiment, measurement noise was wrongly considered as $\sigma_r = 0.5$, $\sigma_\theta = 3.0$ and control noise was rightly considered as $\sigma_v = 0.3m/s$, $\sigma_\gamma = 3^\circ$. The number of particles was 5 (n = 5). Results were obtained over 50 Monte Carlo runs. a) Estimated robot path and estimated landmark with true robot path and true landmark. The dotted line is the estimated path and circles are the estimated landmark positions. b) Estimated pose error with $2-\sigma$ bound.

be clearly seen that the results of the proposed algorithm are better than those of classical FastSLAM. This is because the modified FastSLAM adaptively tuned the measurement covariance matrix R_t, while the covariance matrix measurement R_t in standard FastSLAM is kept fixed over time, as shown in Figure 14. Figure 14 also shows that the measurement covariance matrix measurement R_t converges to the actual covariance matrix R_t in our proposed method.

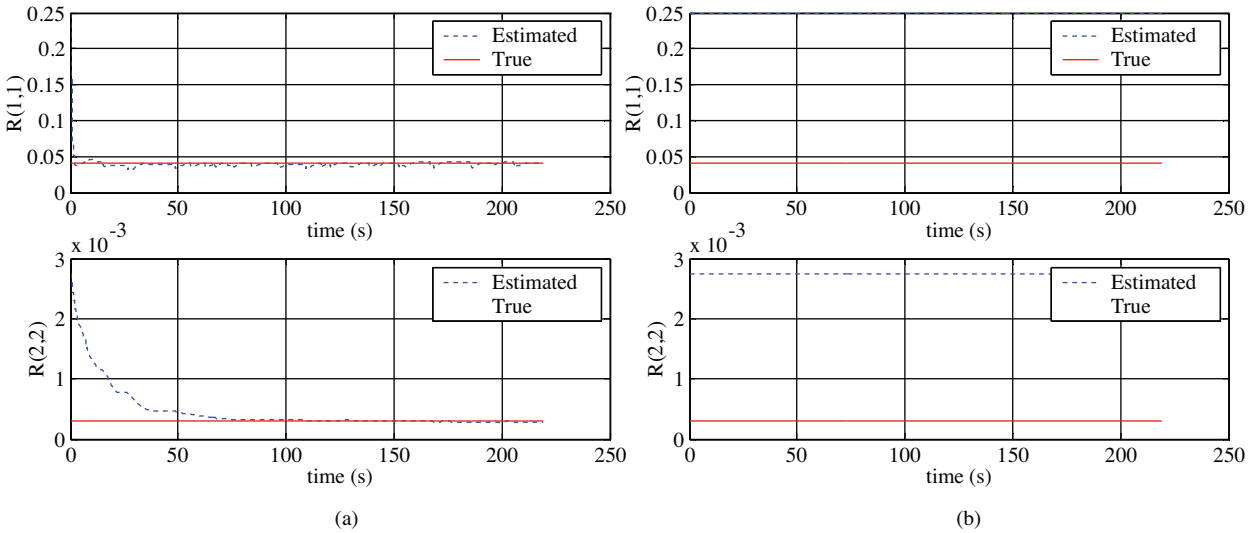

Figure 14. a) Modified FastSLAM, b) FastSLAM.

5. Conclusion

This paper proposed a modified FastSLAM. In the proposed method, 2 problems of FastSLAM are addressed. The first problem is that FastSLAM degenerates over time due to the loss of particle diversity. One of the main reasons for losing particle diversity is sample impoverishment. It occurs when likelihood lies in the tail of the proposal distribution. In this case, most of particle weights are insignificant. Another problem of FastSLAM relates to the design of EKF for the landmark position's estimation. A significant difficulty in designing the EKF can often be traced to incomplete a priori knowledge of the process covariance matrix Q_t and the measurement noise covariance matrix R_t. Incorrect a priori knowledge of Q_t and R_t may seriously degrade the performance of the Kalman filter. This paper presented a modified FastSLAM framework by soft computing. In the proposed method, a neuro-fuzzy EKF for landmark feature estimation and a particle filter based on PSO were presented to overcome the impoverishment of FastSLAM. Experimental results confirmed the effectiveness of the proposed algorithm. The main advantage of our proposed method is its greater consistency compared to classical FastSLAM. This is because, in our proposed method, the theoretical value of the innovation sequence matches its real value. When the motion model is noisier than measurement, the performance of the proposed method also outperforms that of the standard method.

References

[1] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, "A solution to the simultaneous localization and map building (SLAM) problem", IEEE Transactions on Robotics and Automation, Vol. 17, pp. 229-241, 2001.

[2] S. Thrun, D. Fox, W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots", Machine Learning, Vol. 31, pp. 29-53, 1999.

[3] T. Bailey, J. Nieto, E. Nebot, "Consistency of the FastSLAM algorithm", IEEE International Conference on Robotics and Automation, pp. 424-429, 2006.

[4] M. Bosse, J. Leonard, S. Teller, "Large-scale CML using a network of multiple local maps", Workshop Notes of the ICRA Workshop on Concurrent Mapping and Localization for Autonomous Mobile Robots (W4), Washington, DC, 2002.

[5] T. Bailey, Mobile Robot Localization and Mapping in Extensive Outdoor Environments, PhD thesis, University of Sydney, Sydney, NSW, Australia, 2002.

[6] S. Williams, G. Dissanayake, H.F. Durrant-Whyte, "Towards terrain-aided navigation for underwater robotics", Advanced Robotics, Vol. 15, pp. 533-550, 2001.

[7] S. Thrun, D. Hahnel, D. Ferguson, M. Montemerlo, R. Triebel, "A system for volumetric robotic mapping of abandoned mines", in IEEE International Conference on Robotics and Automation, Taipei, Taiwan, pp. 4270-4275, 2003.

[8] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, Cambridge, MIT Press, 2005.

[9] T. Bailey, J. Nieto, E. Nebot, "Consistency of the FastSLAM algorithm", IEEE International Conference on Robotics and Automation, pp. 424-429, 2006.

[10] M. Bolic, P.M. Djuric, S. Hong, "Resampling algorithms for particle filters: a computational complexity perspective", Journal on Applied Signal Processing, Vol. 2004, pp. 2267-2277, 2004.

[11] L. Zhang, X. Meng, Y. Chen, "Convergence and consistency analysis for FastSLAM", IEEE Intelligent Vehicles Symposium, 2009.

[12] M. Montemerlo, FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association, PhD thesis, Carnegie Mellon University, 2003.

[13] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, "FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges", International Joint Conference on Artificial Intelligence, 2003.

[14] N. Kwak, I.K. Kim, H.C. Lee, B.H. Lee, "Adaptive prior boosting technique for the efficient sample size in FastSLAM", in IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, pp. 630-635, 2007.

[15] C. Kim, R. Sakthivel, W.K. Chung, "Unscented FastSLAM: A robust algorithm for the simultaneous localization and mapping problem", in IEEE International Conference on Robotics and Automation, pp. 2439-2445, 2007.

[16] C. Kim, R. Sakthivel, W.K. Chung, "Unscented FastSLAM: a robust and efficient solution to the SLAM problem", IEEE Transactions on Robotics, Vol. 24, pp. 808-820, 2008.

[17] X. Wang, H. Zhang, "A UPF-UKF framework for SLAM", IEEE International Conference on Robotics and Automation, pp. 1664-1669, 2007.

[18] A. Kong, J.S. Liu, W.H. Wong, "Sequential and Bayesian missing data problems", Journal of the American Statistical Association, Vol. 89, pp. 278-288, 1994.

[19] M. Li, B. Hong, R. Luo, Z. Wei, "A novel method for mobile robot simultaneous localization and mapping", Journal of Zhejiang University Science A, pp. 937-944, 2006.

[20] X. Chen, "An adaptive UKF-based particle filter for mobile robot SLAM", International Joint Conference on Artificial Intelligence, 2009.

[21] N. Kwak, G. Kim, B. Lee, "A new compensation technique based on analysis of resampling process in FastSLAM", Robotica, Vol. 26, pp. 205-217, 2007.

[22] I. Kim, N. Kwak, H. Lee, B. Lee, "Improved particle fusing geometric relation between particles in FastSLAM", Robotica, Vol. 27, pp. 853-859, 2009.

[23] R. Havangi, M.A. Nekoui, M. Teshnehlab, "A multi swarm particle filter for mobile robot localization", International Journal of Computer Science Issues, Vol. 7, pp. 15-22, 2010.

[24] R.J. Fitzgerald, "Divergence of the Kalman filter", IEEE Transactions on Automatic Control, Vol. AC-16, pp. 736-747, 1971.

[25] R.K. Mehra, "On the identification of variances and adaptive Kalman filtering", IEEE Transactions on Automatic Control, Vol. AC-15, pp. 175-184, 1970.

[26] R. Havangi, M.A. Nekoui, M. Teshnehlab, "Adaptive neuro-fuzzy extended Kalman filtering for robot localization", International Journal of Computer Science Issues, Vol. 7, pp. 15-23, 2010.

[27] G. Reina, A. Vargas, K. Nagatani, K. Yoshida, "Adaptive Kalman filtering for GPS-based mobile robot localization", IEEE International Workshop on Safety, Security and Rescue Robotics, 2007.

[28] L. Jetto, S. Longhi, "Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots", IEEE Transactions on Robotics and Automation, Vol. 15, pp. 219-229, 1999.

[29] A.P. Sage, G.W. Husa, "Adaptive filtering with unknown prior statistics", Joint Automatic Control Conference, pp. 760-769, 1969.

[30] K. Murphy, "Bayesian map learning in dynamic environments", Proceedings of the Conference on Neural Information Processing Systems, Denver, CO, USA, pp. 1015-1021, 1999.

[31] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association", Journal of Machine Learning Research, Vol. 4, pp. 380-407, 2004.

[32] M. Montemerlo, S. Thrun, D. Koller, B.Wegbreit, "FastSLAM: a factored solution to the simultaneous localization and mapping problem", Proceedings of AAAI National Conference on Artificial Intelligence, Edmonton, AB, Canada, 2002.

[33] G. Grisetti, C. Stachniss, W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposal and selective resampling", IEEE International Conference on Robotics and Automation, pp. 2443-2448, 2005.

[34] G. Grisetti, C. Stachniss, W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters", IEEE Transactions on Robotics, Vol. 23, pp. 34-46, 2007.

[35] R. Krohling, "A Gaussian swarm: a novel particle swarm optimization algorithm", IEEE Conference on Cybernetics and Intelligent Systems, Singapore, pp. 372-376, 2004.

[36] T. Bailey, Source Code, University of Sydney, available at:
http://www-personal.acfr.usyd.edu.au/tbailey/software/index.html, 2008.