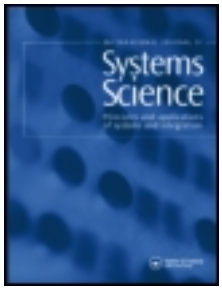


International Journal of Systems Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tsys20

A SLAM based on auxiliary marginalised particle filter and differential evolution

R. Havangi [a], M. A. Nekoui [a], M. Teshnehlab [a] & H. D. Taghirad [a]
[a] Department of Systems and Control, K. N. Toosi University of Technology, Tehran, Iran
Version of record first published: 24 Jan 2013.

To cite this article: R. Havangi, M. A. Nekoui, M. Teshnehlab & H. D. Taghirad (2013): A SLAM based on auxiliary marginalised particle filter and differential evolution, International Journal of Systems Science, DOI:10.1080/00207721.2012.759299

To link to this article: http://dx.doi.org/10.1080/00207721.2012.759299

International Journal of Systems Science, 2013
http://dx.doi.org/10.1080/00207721.2012.759299

Taylor & Francis
Taylor & Francis Group

A SLAM based on auxiliary marginalised particle filter and differential evolution

R. Havangi*, M.A. Nekoui, M. Teshnehlab and H.D. Taghirad

Department of Systems and Control, K. N. Toosi University of Technology, Tehran, Iran

(*Received 15 November 2011; final version received 11 December 2012*)

FastSLAM is a framework for simultaneous localisation and mapping (SLAM) using a Rao-Blackwellised particle filter. In FastSLAM, particle filter is used for the robot pose (position and orientation) estimation, and parametric filter (i.e. EKF and UKF) is used for the feature location's estimation. However, in the long term, FastSLAM is an inconsistent algorithm. In this paper, a new approach to SLAM based on hybrid auxiliary marginalised particle filter and differential evolution (DE) is proposed. In the proposed algorithm, the robot pose is estimated based on auxiliary marginal particle filter that operates directly on the marginal distribution, and hence avoids performing importance sampling on a space of growing dimension. In addition, static map is considered as a set of parameters that are learned using DE. Compared to other algorithms, the proposed algorithm can improve consistency for longer time periods and also, improve the estimation accuracy. Simulations and experimental results indicate that the proposed algorithm is effective.

Keywords: FastSLAM; auxiliary marginal particle filter; differential evolution

1. Introduction

The simultaneous localisation and mapping (SLAM) is a key issue to achieve intelligent navigation for mobile robots. The two key computational solutions to SLAM are the extended Kalman filter (EKF-SLAM) and the Rao-Blackwellised particle filter (FastSLAM). The EKF-SLAM is the most popular approach to solve the SLAM problem. However, EKF-SLAM suffers from problems such as the computational complexity and data association (Dissanayake, Newman, Clark, and Durrant-Whyte 2001; Huang and Dissanayake 2007; Hua and Shi-Yin 2011). To solve the problems of EKF-SLAM, the FastSLAM algorithm is proposed (Montemerlo, Thrun, Koller, and Wegbreit 2003; Thrun et al. 2004). FastSLAM is an instance of Rao-Blackwellised particle filter, which partitions the SLAM posterior into a localisation problem with an independent landmark position estimation problem (Montemerlo et al. 2003; Thrun et al. 2004). However, it has been recently reported that FastSLAM is inconsistent (Bailey, Nieto, and Nebot 2006; Kim, Sakthivel, and Chung 2007; Kwak, Kim, and Lee 2007a; Kwak, Kim, Lee, and Lee 2007b; Kim, Sakthivel, and Chung 2008; Kim, Kwak, Lee, and Lee 2009). It is shown that FastSLAM degenerates with time, regardless of the density of landmarks and the number of particles within an environment, and will always produce optimistic estimates of uncertainty in the long term. Therefore, FastSLAM is unable to adequately explore state space to be a reasonable Bayesian estimator.

Many approaches have been experienced to improve FastSLAM. In Kim et al. (2009), improved particle filter using geometric relation between particles is proposed to restrain particle depletion and to reduce estimation errors and error variances. In this approach, KD tree (k-dimensional tree) is used in the resampling process to restrain particles to go awry from the real pose of the robot. In Kwak et al. (2007a), various resampling algorithms have been analysed using the computer simulations. Moreover, a new compensation technique has been proposed instead of resampling to resolve the particle depletion problem. A few attempts use evolutionary algorithms to improve FastSLAM. In Lee, Park, Choi, and Lee (2009), an improved FastSLAM framework using particle swarm optimisation (PSO), named PSO-FastSLAM, is presented. In this work, the concept of particle cooperation in swarm intelligence is used to improve the performance of FastSLAM. In Kim et al. (2008), UFastSLAM is presented. In this approach, as the vehicle and the feature states are estimated without accumulating linearisation errors, the accuracy of the state estimation has been improved over the previous approaches. In Rodriguez-Losada, San Segundo, Matia, and Pedraza (2009), a dual version of FastSLAM is presented. This approach decouples SLAM into a map estimation and a localisation problem, using a particle filter to estimate over maps and a Kalman filter attached to each particle to estimate the robot pose conditioned to the given map. In all these algorithms, the presence of a static parameter (i.e. static map) in the state space prevents the particle

*Corresponding author. Email: havangi@kntu.ac.ir

Table 1. Comparison of algorithms.

Algorithm	Robot pose	Feature position
FastSLAM2.0	The robot path is estimated by extended particle filter	EKF
UFastSLAM	The robot path is estimated by unscented particle filter	UKF
Proposed method	The robot pose is estimated by auxiliary marginalised particle filter	DE

approximation to be converged uniformly in time (Crisan and Doucet 2002). To overcome this problem, marginal SLAM is introduced in Martinez-Cantin, de Freitas, and Castellanos (2007). However, this technique suffers from sample impoverishment and high computational cost. The sample impoverishment occurs when prior distribution is either much broader than the likelihood or most measurements appear in the tail of the proposal distribution. In this case, the weights of most participles are insignificant. Furthermore, the computational complexity of marginal SLAM is $O(M^2)$, where M is the number of particles used to represent the probability density function.

In this paper, SLAM based on auxiliary marginalised particle filter and DE is proposed. In summary, Table 1 shows comparison between the proposed method with FastSLAM2.0 and UFastSLAM. In this approach, the robot pose is estimated using auxiliary marginal particle filter that operates directly on the marginal distribution. This technique avoids performing importance sampling on a space of growing dimension. In addition, static map is considered as parameters that are estimated by evolutionary computation. For this purpose, the map estimation is considered as an optimisation problem. In general, this optimisation problem has no explicit analytical solution. By employing the conventional methods, a local solution is usually obtained. To solve this problem, in this study, an evolutionary computation procedure is used. Although there are many different types of evolutionary algorithms, DE is used to solve the problem. This is because DE is easily implementable and has few parameters to be adjusted. Furthermore, in DE, particles cooperate with each other.

The rest of the paper is organised as follows. In Section 2, the SLAM problem and required background are reviewed. The proposed algorithm is presented in Section 3. In Section 4, the simulation and experimental results are shown.

2. Background

2.1. FastSLAM

To describe SLAM, let us denote the map by Θ and the robot pose at time t by s_t. The map Θ consists of a collection of features, each of which will be denoted by θ_n and the total number of stationary features will be denoted by N. In this situation, the SLAM problem can be formulised in a Bayesian probabilistic framework as

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

where $s^t = \{s_1, \ldots, s_t\}$ is the robot path, $z^t = \{z_1, \ldots, z_t\}$ is a sequence of measurements, $u^t = \{u_1, \ldots, u_t\}$ is a sequence of control inputs and $n^t = \{n_1, \ldots, n_t\}$ is the data association, in which each n_t specifies the identity of the landmark observed at time t. FastSLAM is an efficient algorithm for the SLAM problem that is based on a straightforward factorisation as follows (Kim et al. 2007; Kwak et al. 2007a, 2007b; Kim et al. 2009):

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

This factorisation states that the SLAM problem can be decomposed into estimating the product of a posterior over the robot path and N landmark posteriors given the knowledge of the robot path. The FastSLAM algorithm implements the robot path posterior $p(s^t | z^t, u^t, n^t)$ using a particle filter, and the landmark posteriors $p(\theta_n | s^t, z^t, u^t, n^t)$ are realised using parametric filter. Each particle in FastSLAM is denoted as

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

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 $s^{t,[m]}$. Since it is usually impossible to sample from the true posterior, the samples are generated from a proposal distribution as (Kwak et al. 2007a):

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

The importance weights can be updated in time as

$$w_t^{[m]} = \frac{p(s^{t,[m]} | z^t, u^t, n^t)}{q(s^{t,[m]} | z^t, u^t, n^{t-1})}. \tag{5}$$

One of the main reasons for losing consistency is related to the structure of FastSLAM. As the joint posterior density of the state is approximated using sequential importance sampling, the dimension of the target density $p(s^t | z^t, u^t, n^t)$ grows with each time step. This causes the algorithm to be degenerated quickly and therefore use of resampling strategies becomes necessary in order to ensure a reasonable approximation of the target density.

In a formal manner, the resampling step should be done along the full path $(s^{t,[m]}, \omega^{t,[m]})$ that increases the number of particles as time t goes on to maintain the desired level of accuracy. In implementation of FastSLAM, all authors carry out resampling over the marginal space $(s_t^{[m]}, \omega_t^{[m]})$. This would be good enough if the system exhibits 'exponential forgetting' of its past errors (Crisan and Doucet 2002). However, static map estimate depends on the whole state trajectory and does not necessarily exhibit an exponential forgetting behaviour. The resampling of trajectories in the joint path space is guaranteed to deplete the past in finite time. However, as the number of particles increases exponentially, the system becomes intractable. Moreover, the resampling step in marginal space provides an inconsistent estimation with a finite number of particles as the system does not exhibit 'exponential forgetting' of its past errors.

2.2. *Differential evolution*

Differential evolution (DE) is a stochastic, population-based search strategy (Engelbrecht 2007). While DE shares similarities with other evolutionary algorithms, it differs mainly in the sense that distance and direction information from the current population is used to guide the search process. The DE algorithm utilises N_p parameter vectors $\{x_i^k, i = 1, \ldots, N_p\}$ as a population at iteration step k. The parameter vector is denoted by $x_i = [x_{i1}, x_{i2}, \ldots, x_{in_d}]$ with components x_{ij}. The index $i = 1, 2, \ldots, N_p$ represents the individual's index in the population and $j = 1, 2, \ldots, n_d$ is the position in D-dimensional individual, where n_d is the problem dimension. During the initialisation of the population, N_p vectors are generated randomly in the D-dimensional search space. After initialisation, mutation and crossover operators are employed to generate new candidate vectors, and a selection scheme is applied to determine whether the offspring or the parent survives to the next generation. The above process is repeated until a final criterion is reached.

2.2.1. *Mutation operator*

The DE mutation operator produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential term. For each parent $x_i(t)$, the trial vector $u_i(t)$ is created as (Engelbrecht 2007)

$$u_i(t) = x_{i_1}(t) + \beta(x_{i_2}(t) - x_{i_3}(t)), \quad i \neq i_1 \neq i_2 \neq i_3 \tag{6}$$

with randomly chosen indices, $i_1, i_2, i_3 \in \{1, 2, \ldots, N_P\}$. Parameter β is the scaling factor and its value is within the range $(0, \infty)$. It controls the speed and robustness of the search.

2.2.2. *Crossover operator*

The DE crossover operator implements a discrete recombination of the trial vector $u_i(t)$ and the parent vector $x_i(t)$ to produce offspring $x_i'(t)$. The crossover is implemented as follows (Chang, Lu, and Wang 2004):

$$x_{ij}'(t) = \begin{cases} u_{ij}(t) & \text{if } j \in J \\ x_{ij}(t) & \text{otherwise} \end{cases}, \tag{7}$$

where J is the set of crossover points. For binomial crossover, crossover points are randomly selected from the set of possible crossover points, $\{1, 2, \ldots, n_d\}$.

2.2.3. *Selection operator*

This operation selects a better individual candidate between the parent vector and the trial vector according to their cost value calculated by means of the objective function.

3. SLAM based on auxiliary marginalised particle filter and differential evolution

This section presents a novel SLAM framework based on marginal particle filter and DE. To demonstrate this approach, we reformulate the SLAM problem as a robot localisation problem with unknown observation model parameters. For this purpose, the robot pose s_t is modelled with a Markov process, which is characterised by an initial distribution of initial distribution $p(s_t)$ and transition prior $p(s_t|s_{t-1}, u_t)$, which represents the motion model. The observations z_t are assumed to be conditionally independent given the process s_t and marginal distribution $p_{\theta_t}(z_t|s_t)$, where θ_{n_t} is a vector describing the elements of the map (or parameters) in time t. Hence, the model consists of the following two distributions (Martinez-Cantin et al. 2007):

$$p(s_t|s_{t-1}, u_t) \tag{8}$$

$$p_{\theta_{n_t}}(z_t|s_t). \tag{9}$$

From this viewpoint, the SLAM problem is to compute sequentially in time the filtering distribution $p_{\theta_{n_t}}(s_t|z_{1:t})$ and point estimates of the map θ_{n_t}. The estimation of both the dynamic state and static parameters is usually known as the dual estimation in literatures. Many researchers have tried to solve this problem in the field of filtering for simple case that the number of parameter is fixed. The application of particle filter for recursive state and parameter estimation of a simulated batch polymerisation reactor is presented in Chen, Morris, and Martin (2005). In Doucet and Tadic (2003) and Poyiadjis, Doucet, and Singh (2005), gradient-based maximum likelihood estimation technique is presented, where a particle filter is used to numerically approximate the likelihood function derivatives. In Kantas,

Doucet, Singh, and Maciejowski (2009), a comprehensive review of the state of the art on joint state and parameter estimation techniques based on particle filter is introduced. Several authors have proposed the technique of expectation maximisation algorithm combined with the particle filter for combined state and parameter estimation of nonlinear systems (Wills, Schon, and Ninness 2008).

In this paper, a dual estimation algorithm is proposed for SLAM problem where the number of parameters increases over time. This algorithm combines DE with marginal particle filtering that operates directly on the marginal distribution. This is because that marginal particle filtering is superior over the particle filter in terms of the importance weight variance (Klaas, de Freitas, and Doucet 2005). In the proposed method, provided that the robot pose s_t could be estimated using marginal particle filtering then we can access the set of pairs $\{s_1, z_1\}, \ldots, \{s_t, z_t\}$ and compute the parameters using DE of those pairs. The proposed algorithm consists of the vehicle state estimation and map learning that will be illustrated in the following subsections.

3.1. Vehicle state estimation

In the proposed approach, marginal particle method is used to numerically approximate the filtering distribution. The predictive density is obtained by marginalising as

$$p_{\theta_{n_t}}(s_t|z_{1:t-1}) = \int p(s_t|s_{t-1}) p_{\theta_{n_t}}(s_{t-1}|z_{1:t-1}) ds_{t-1}. \quad (10)$$

Hence, the filtering update becomes

$$\begin{aligned} p_{\theta_{n_t}}(s_t|z_{1:t}) &= p_{\theta_{n_t}}(z_t|s_t) p_{\theta_{n_t}}(s_t|z_{1:t-1}) \\ &= p_{\theta_{n_t}}(z_t|s_t) \int p(s_t|s_{t-1}) p_{\theta_{n_t}}(s_{t-1}|z_{1:t-1}) ds_{t-1}. \end{aligned} \quad (11)$$

The integral in Equation (11) is generally not solvable analytically, but since we have a particle approximation of $p_{\theta_{n_t}}(s_{t-1}|z_{1:t-1})$(namely,$\{x_{t-1}^{[i]}, w_{t-1}^{[i]}\}$); Equation (10) can be approximated as follows:

$$p_{\theta_{n_t}}(s_t|z_{1:t-1}) = \sum_{i=1}^{N} \omega_{t-1}^{[i]} p(s_t|s_{t-1}^{[i]}). \quad (12)$$

.To draw samples from $p_{\theta_{n_t}}(s_t|z_{1:t})$, a proposal distribution is chosen. The choice of the proposal distributions is one of the most critical issues in the design of SLAM based on particle filter. The most common strategy is to sample from the transition motion (or prior distribution). This strategy is effective when the observation accuracy is low. However, this strategy can fail when prior distribution is much broader than the likelihood. In this case, sample impoverishment occurs. The optimal choice of the proposal distribution is the posterior distribution:

$$q_{\theta_{n_t}}(s_t|z_{1:t}) = p_{\theta_{n_t}}(s_t|z_{1:t}) = p_{\theta_{n_t}}(z_t|s_t) \sum_{i=1}^{N} \omega_{t-1}^{[i]} p\left(s_t|s_{t-1}^{[i]}\right). \quad (13)$$

However, it is often hard to obtain samples from this proposal distribution. By using the auxiliary marginal particle filter in Klaas et al. (2005), it is possible to obtain samples from a proposal distribution that is close to the posterior distribution. The posterior distribution can equivalently be written as

$$\begin{aligned} p_{\theta_{n_t}}(s_t|z_{1:t}) &= p_{\theta_{n_t}}(z_t|s_t) \sum_{i=1}^{N} \omega_{t-1}^{[i]} p(s_t|s_{t-1}^{[i]}) \\ &= \sum_{i=1}^{N} \omega_{t-1}^{[i]} p_{\theta_{n_t}}(z_t|s_{t-1}^{[i]}) p(s_t|s_{t-1}^{[i]}, z_t) \\ &= \sum_{i=1}^{N} p(i|z_{1:k}) p\left(s_t|s_{t-1}^{[i]}, z_t\right), \end{aligned} \quad (14)$$

where $p(i|z_{1:k})$ is as

$$\begin{aligned} p(i|z_{1:k}) &\propto \omega_{t-1}^{[i]} p_{\theta_{n_t}}\left(z_t|s_{t-1}^{[i]}\right) \\ &= \omega_{t-1}^{[i]} \int p_{\theta_{n_t}}(z_t|s_t) p\left(s_t|s_{t-1}^{[i]}\right) ds_t. \end{aligned} \quad (15)$$

Using Equation (14), we can sample from the proposal distribution $p(i|z_{1:k})$ at the first by sampling i, which is known as the auxiliary variable from $\hat{p}(i|z_{1:k})$, and then sampling s_t from $p(s_t|s_{t-1}^{[i]}, z_t)$. Since Equation (15) cannot usually be evaluated as analytically, we approximate $\hat{p}(i|z_{1:k})$ as

$$\hat{p}(i|z_{1:k}) \propto \omega_{t-1}^{[i]} p_{\theta_{n_t}}\left(z_t|\mu_t^{[i]}\right), \quad (16)$$

where

$$\mu_t^{[i]} = E\left(s_t|s_{t-1}^{[i]}\right). \quad (17)$$

Since sampling from $p(s_t|s_{t-1}^{[i]}, z_t)$ is not always possible, we sample from $p(s_t|s_{t-1}^{[i]})$ instead. Hence, the proposal distribution of auxiliary marginal particle filter is as

$$q_{\theta_{n_t}}(s_t|z_{1:t}) \propto \sum_{i=1}^{N} \hat{p}(i|z_{1:k}) p\left(s_t|s_{t-1}^{[i]}\right). \quad (18)$$

In summary, the pseudo-code description of the robot pose estimation using auxiliary marginal particle filter is as follows:

(1) For $i = 1, \ldots, N$, choose simulation $\mu_t^{[i]}$ and calculate mixture weights:

$$\mu_t^{[i]} = E\left(s_t | s_{t-1}^{[i]}\right)$$
$$\hat{\lambda}_{t-1}^{[i]} = \omega_{t-1}^{[i]} p_{\theta_{n_t}}\left(z_t | \mu_t^{[i]}\right)$$
$$\lambda_{t-1}^{[i]} = \frac{\hat{\lambda}_{t-1}^{[i]}}{\sum_{i=1}^{N} \hat{\lambda}_{t-1}^{[i]}}.$$

(2) For $i = 1, \ldots, N$, sample from the proposal distribution:

$$q_{\theta_{n_t}}(s_t | z_{1:t}) \sim \sum_{i=1}^{N} \lambda_{t-1}^{[i]} p\left(s_t | s_{t-1}^{[i]}\right).$$

(3) For $i = 1, \ldots, N$, evaluate the importance weights:

$$\tilde{w}_t^{[i]} = \frac{p_{\theta_{n_t}}\left(z_t | s_t^{[i]}\right) \sum_{j=1}^{N} w_{t-1}^{[j]} p\left(s_t^{[i]} | s_{t-1}^{[j]}\right)}{\sum_{j=1}^{N} \lambda_{t-1}^{[j]} p\left(s_t^{[i]} | s_{t-1}^{[j]}\right)}.$$

(4) Normalise the importance weights,

$$w_t^{[i]} = \frac{\tilde{w}_t^{[i]}}{\sum_{j=1}^{n} \tilde{w}_t^{[j]}}.$$

3.3. *Learning map*

Since static map has been considered as a set of parameters, the aim of mapping is to identify these parameters based on an observation sequence $z_{1:t}$ in an online fashion using maximum likelihood. When feature is not observed, the position feature remains unchanged. The observed feature is learned using maximum likelihood. The maximum likelihood would lead to find the maximum of likelihood function $p_{\theta_{n_t}}(z_{0:t})$ with respect to θ_{n_t} as follows:

$$p_{\theta_{n_t}}(z_{0:t}) = p_{\theta_{n_t}}(z_t | z_{0:t-1}) p_{\theta_{n_t}}(z_{0:t-1}) = \prod_{k=0}^{t} p_{\theta_{n_t}}(z_k | z_{0:k-1}). \tag{19}$$

As a result, the problem of learning map can be easily stated as an optimisation problem. This optimisation problem can be reformulated in an equivalent and convenient form by taking logarithms as follows:

$$l_{\theta_{n_t}}(z_{0:t}) = \log(p_{\theta_{n_t}}(z_{0:t})) = \sum_{k=0}^{t} \log p_{\theta_{n_t}}(z_k | z_{1:k-1}). \tag{20}$$

The maximum likelihood estimation can be defined as

$$\theta_{n_t} = \arg\max_{\theta_{n_t}} l_{\theta_{n_t}}(z_{0:t}) = \arg\max_{\theta_{n_t}} \sum_{k=0}^{t} \log p_{\theta_{n_t}}(z_k | z_{1:k-1}). \tag{21}$$

To implement this optimisation, a recursive formulation is required. If we consider the objective function as $f_o(s_t)$, it can be expressed recursively as follows:

$$f_o(t) = \sum_{k=0}^{t} \log p_{\theta_{n_t}}(z_k | z_{1:k-1}) = \log p_{\theta_{n_t}}(z_t | z_{1:t-1}) + \sum_{k=0}^{t-1} \log p_{\theta_{n_t}}(z_k | z_{1:k-1}). \tag{22}$$

On the other hand,

$$f_o(t-1) = \sum_{k=0}^{t-1} \log p_{\theta_{n_t}}(z_k | z_{1:k-1}). \tag{23}$$

Therefore,

$$f_o(t) = \log p_{\theta_{n_t}}(z_t | z_{1:t-1}) + f_o(t-1). \tag{24}$$

Using Bellman's principle of optimality, the optimisation problem can be reformulated as

$$f(\theta) = \log p_{\theta_{n_t}}(z_t | z_{1:t-1}), \tag{25}$$

where

$$p_{\theta_{n_t}}(z_t | z_{0:t-1}) = \int p_{\theta_{n_t}}(z_t | s_{t-1}) p_{\theta_{n_t}}(s_{t-1} | z_{0:t-1}) ds_{t-1}. \tag{26}$$

Given the particle approximation of $p_{\theta_{n_t}}(s_{t-1} | z_{1:t-1})$, Equation (26) is calculated as

$$p_{\theta_{n_t}}(z_t | z_{0:t-1}) = \sum_{i=1}^{N} \omega_{t-1}^{[i]} p\left(z_t | \tilde{s}_t^{[i]}\right). \tag{27}$$

On the other hand, the probability $p_{\theta_{n_t}}(z_t | \tilde{s}_t^{[i]})$ can be expressed as

$$p_{\theta_{n_t}}(z_t | \tilde{s}_t^{[i]}) = \frac{1}{2\pi\ |R|^{1/2}} e^{-\frac{1}{2}(z_t - \hat{z}_t^{[i]}) R^{-1} (z_t - \hat{z}_t^{[i]})^T}. \tag{28}$$

By replacing the expressions of $p_{\theta_{n_t}}(z_t | \tilde{s}_t^{[i]})$ in Equation (28), the optimisation problem can be reformulated as

$$\arg\min_{\theta_{n_t}} \sum_{j=1}^{N} \left(z_t - \hat{z}_t^{[j]}\right) R^{-1} \left(z_t - \hat{z}_t^{[j]}\right)^T. \tag{29}$$

Hence, the learning map is reformulated as an optimisation problem. To solve this problem, the DE algorithm is used. DE uses a vector $\theta^k_{n_t,i}$ to represent each candidate solution i to the optimisation problem at iteration k for a given time step t. DE will generate a new set of trail vector u as

$$u_t = \theta^k_{n_t,i} + \beta(\theta^k_{n_t,i_2} - \theta^k_{n_t,i_3}), \tag{30}$$

where $\theta^k_{n_t,i}$ is the target vector to be perturbed at iteration k, $\theta^k_{n_t,i_2}$ and $\theta^k_{n_t,i_3}$ are randomly selected two individuals from population so that i_2 and i_3 are different from running index i. To increase the diversity of the new generation of vectors, a crossover mechanism is introduced as follows:

$$\theta'_{n_t,ij} = \begin{cases} u_{t,ij} & \text{if j} \in \text{J} \\ \theta_{n_t,ij} & \text{otherwise} \end{cases}. \tag{31}$$

To decide whether or not vector θ'_{n_t} should become a member of generation $t+1$, the new vector is compared to θ_{n_t}. If vector θ'_{n_t} yields a better value for the fitness function than θ_{n_t}, θ_{n_t} is replaced by θ'_{n_t} for the new generation; otherwise, the old value θ_{n_t} is retained for the new generation. Because DE is an iteration algorithm in search space, it will cost much time. However, DE in our algorithm does not need iterate so many times. This is because searching space is a small area around the position at time step $t-1$ and initialised particles are also around the true position. When the best fitness value reaches a certain threshold, the iteration is stopped. In summary, the DE algorithm for learning each observed feature θ_{n_t} consists of the following steps:

Step 1: Population initialisation of C
Generate N chromosomes randomly from a feasible range.
Step 2: Evolution phase

(a) Take measurements to the landmarks and perform data association.
(b) Create the trial vector, u_t by applying the mutation operator.
(c) For each θ'_{n_t}, the expected observations are calculated and the fitness function is evaluated. If $f(\theta'_{n_t})$ is better than $f(\theta_{n_t})$, then add θ'_{n_t} to $C(t+1)$, otherwise the original θ_{n_t} adds to $C(t+1)$.
(d) The chromosome of lower value of the fitness function is selected as the best landmark estimate. Go to step 2(b) and begin a new iteration.

Step 3: Updating
The best chromosome of the population is used as the updated landmark estimate.

In summary, the pseudocode of the proposed method is as follows:

For all particles

Predict mean and covariance of the vehicle
For all observations
$\hat{n}$ = DataAssociation()

For $\hat{n}$ = known feature

Sample a new robot pose

$$s_t^{[i]} \sim \sum_{i=1}^{N} \lambda_{t-1}^{[i]} p\left(s_t | s_{t-1}^{[i]}\right)$$

Calculate important weight

$$\tilde{w}_t^{[i]} = \frac{p_{\theta_{n_t}}\left(z_t | s_t^{[i]}\right) \sum_{j=1}^{N} w_{t-1}^{[j]} p\left(s_t^{[i]} | s_{t-1}^{[j]}\right)}{\sum_{j=1}^{N} \lambda_{t-1}^{[j]} p\left(s_t | s_{t-1}^{[j]}\right)}$$

End for

Normalise the importance weights

$$w_t^{[i]} = \frac{\tilde{w}_t^{[i]}}{\sum_{j=1}^{n} \tilde{w}_t^{[j]}}$$

Calculate feature mean

If $\hat{n}$ = known feature

Feature estimation by DE

else

Calculate new feature mean

End if
For unobserved features

$$\theta_{n_t} = \theta_{n_{t-1}}$$

End for
End for

3.4. Computational complexity

The computational complexity of the proposed algorithm is related to the sampling strategy, the learning map and the calculation of the importance weight. Hence the number of particles, M, the number of features, N, and the

Table 2. Comparison of complexity computational.

Operation	Complexity	
	Proposed method	UFastSLAM and FastSLAM
Sampling	O(M)	O(M)
Feature estimation	O(M)	O(M)
Importance weight	O(M)	O(M)

number of iterations in DE, k, determine the complexity of it. The complexity of computing the proposal distribution and calculating the importance weights is O(M^2). This computational complexity can be reduced to $O(M)$ using improved fast Gaussian transform (IFGT) (see Appendix for more details). In addition, the learning map has a complexity of O(KM) that is approximated to O(M). This is because DE does not need to iterate so many times, as mentioned previously. Table 2 depicts the complexity of the individual operations in algorithms. As can be seen, the proposed approach has a complexity, which is almost the same as that of FastSLAM and UFastSLAM.

4. Results

Since the proposed algorithm incorporates the current observation into the proposal distribution, the performance of it is superior to marginal SLAM algorithm in nearly all aspects. Hence, the performance of the proposed method is compared with methods that measurements incorporate in the proposal distribution (i.e. FastSLAM2.0 and UFastSLAM). The performance of algorithms is evaluated on simulated data and real world data sets. The simulated data allow comparison with ground truth, while the real world data prove the applicability of algorithms to practical problems.

4.1. Simulation results

Simulation has been carried out to evaluate the performance of the proposed approach in comparison with other algorithms for the benchmark environment, available in http://www.personal.acfr.usyd.edu.au/tbailey/software/index.html. Figure 1 shows the robot trajectory and landmark location. The star points (*) depict the location of the landmarks that are known and stationary in the environment.

The robot has 4 m wheelbase and is equipped with a range-bearing sensor with a maximum range of 30 m and a 180° frontal field of view. The robot moves at a speed 3 m/s and with a maximum steering angle of 30°. The control noise is ($\sigma_v = 0.2$ m/s,$\sigma_\gamma = 2^o$) and the measurement noise

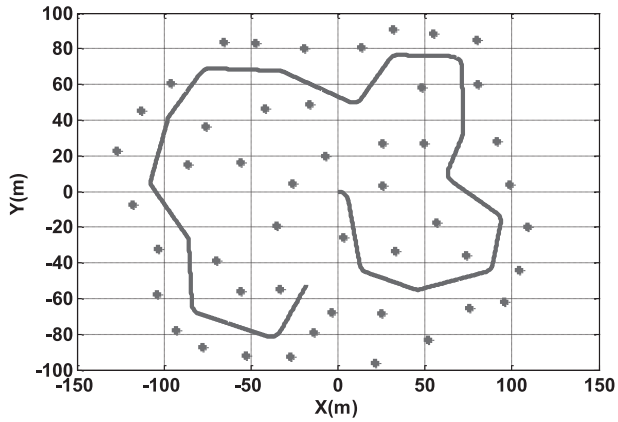


Figure 1. The experiment environment.

is ($\sigma_r = 0.1$ m, $\sigma_\gamma = 0.1^o$). Control frequency is 40 Hz and observation scans are obtained at 5 Hz.

4.1.1. Performance comparison of algorithms

Figure 2 shows the comparison between the proposed algorithm and other algorithms. The results are obtained over 20 Monte Carlo runs. It can be clearly seen that the results of the proposed algorithm are better than that of UFastSLAM and FastSLAM2.0. In other words, the estimated robot path and landmark positions with their actual value coincide as closely as possible. The root mean square error (RMSE) of the algorithms is shown in Figure 3. The mean and variance of RMSE are calculated over 20 independent runs with 30 particles for each algorithm. Each bar in this figure represents the mean and variance of RMSE of the robot position. As shown, the mean and variance of RMSE of the proposed method is smaller than that of UFastSLAM and FastSLAM2.0 and, as expected, the accuracy of UFastSLAM is much better than that of FastSLAM2.0. The performance of algorithms is compared with various numbers of particles. Figure 4 shows results for this case. It is observed that the performance of the proposed algorithm does not depend heavily on the number of particles.

4.1.2. Consistency of algorithms

To verify the consistency of algorithms, average normalised estimation error squared (NEES) is used. For an available ground truth x_k and an estimated mean and covariance $\{\hat{x}, \hat{P}\}$, we can use NEES to characterise the filter performance (Bar-Shalom, Li, and Kirubarajan 2001):

$$\varepsilon_k = (x_x - \hat{x}_k)^T P_k^{-1} (x_x - \hat{x}_k). \tag{32}$$

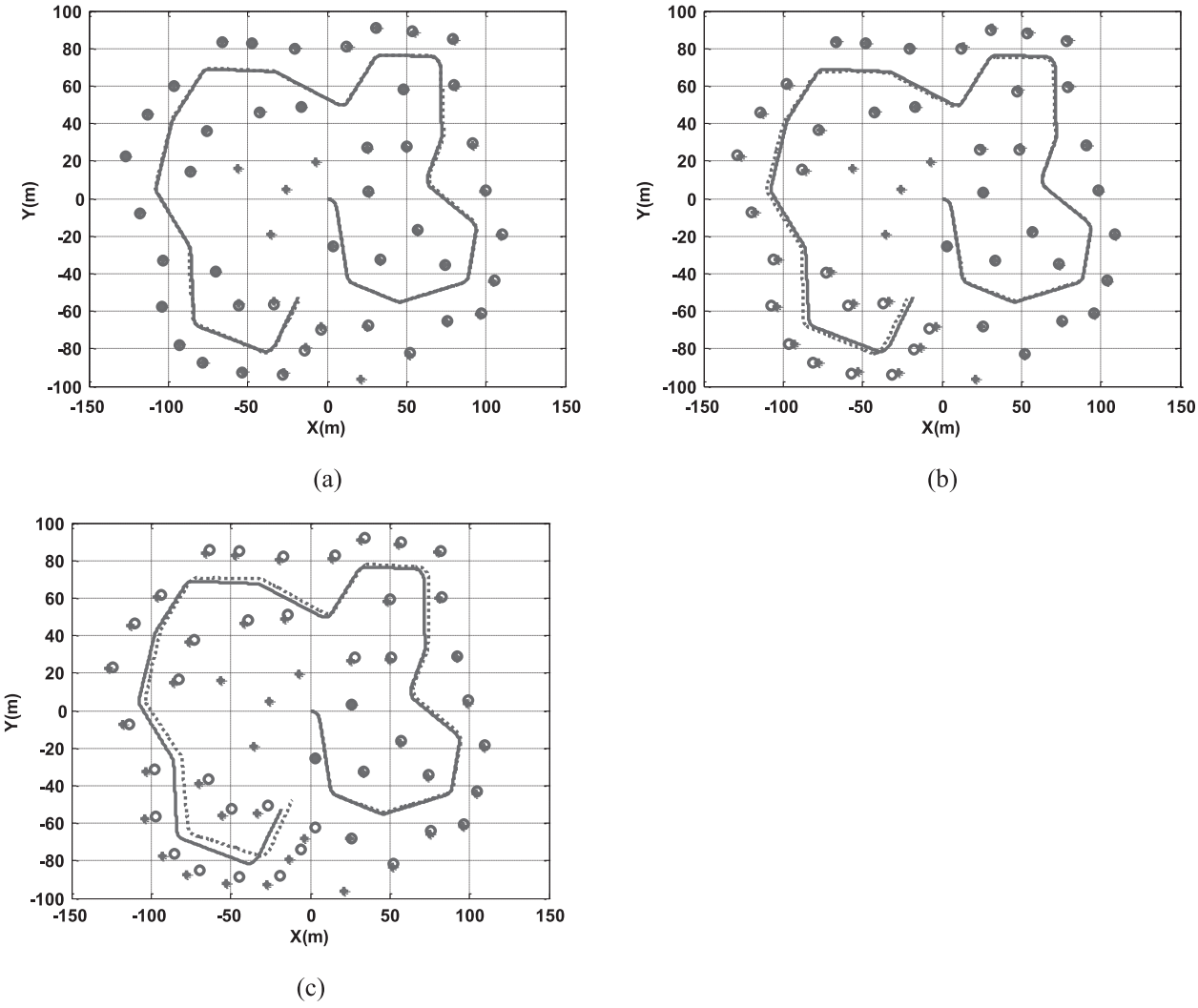


Figure 2. Estimated and true vehicle paths with estimated and true landmarks: (a) Proposed method, (b) UFastSLAM and (c) FastSLAM2.0. The red line and red stars denote the true path and landmark positions, respectively. The blue line is the mean estimate of vehicle position and the blue circles are estimated landmarks.

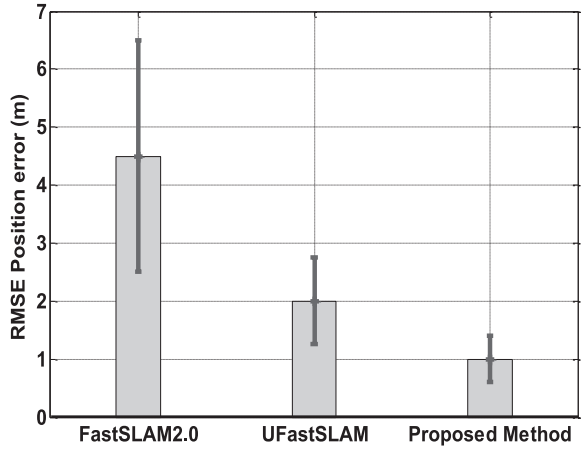


Figure 3. RMSE of the robot position.

Consistency is evaluated by performing multiple Monte Carlo runs and computing the average NEES. Given N runs, the average NEES is computed as (Bar-Shalom et al. 2001)

$$\bar{\varepsilon}_k = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_{ik}. \tag{33}$$

Assuming a consistent linear-Gaussian filter, $N_R\bar{\varepsilon}_t$ has a χ^2 density with $N_R \dim(s_t)$ degrees of freedom. Thus, for the two-dimensional robot position, using 20 Monte Carlo simulations, the two-sided 95% probability concentration region for $\bar{\varepsilon}_k$ is bounded by interval [1.3, 2.79]. Figure 5 shows the consistency of the proposed method in comparison with UFastSLAM and FastSLAM2.0. Since FastSLAM

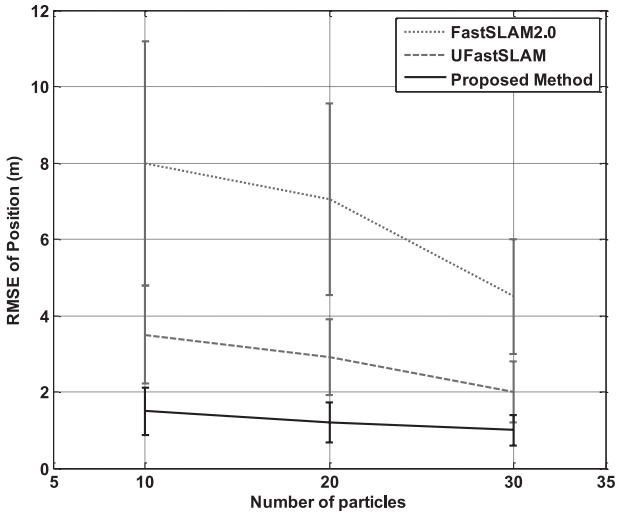
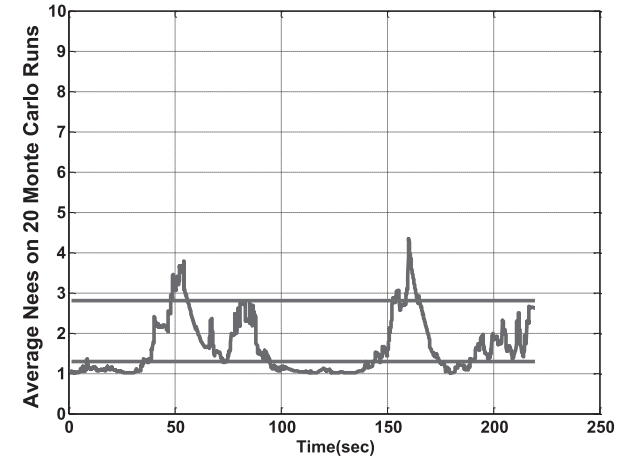


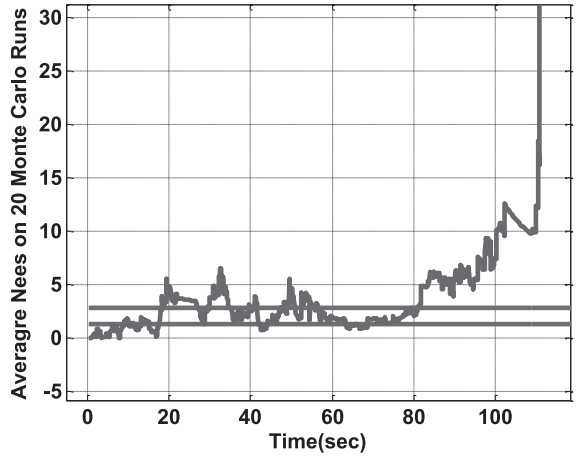
Figure 4. RMSE of algorithms with different numbers of particles.

2.0 and UFastSLAM are inconsistent after 80 s and 35 s, respectively, we zoom in only on this part of graphs as shown in Figure 5(b) and (c). The result shows that consistency of the proposed method is more than that of UFastSLAM and FastSLAM2.0. This occurs because the diversity of particles is more than that of other algorithms. The diversity of particles is analysed through the percentage of particles that do not share an ancestor in time. Every time a particle survives a resampling step with one or more copies of it, the original particle is considered the ancestor of all those copies.

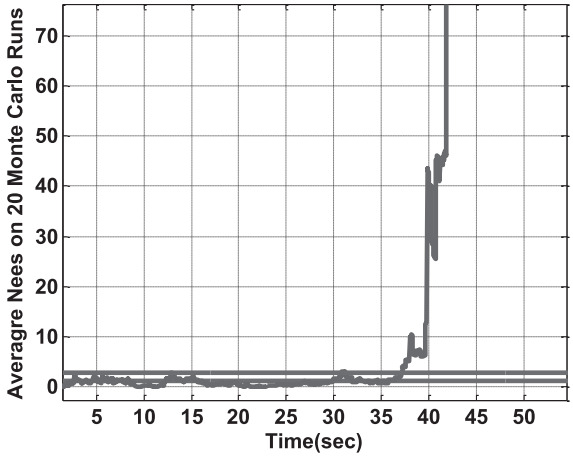
Figure 6 shows that the number of distinct particles in the proposed method is more than that of other algorithms. Moreover, as all algorithms use the same resampling strategy, the particle diversity of UFastSLAM is mostly the same as FastSLAM2.0. As a result, the consistency of the proposed method is more than that of other algorithms.



(a)



(b)



(c)

Figure 5. (a) Proposed method, (b) UFastSLAM (zoomed figure) and (c) FastSLAM2.0 (zoomed figure).

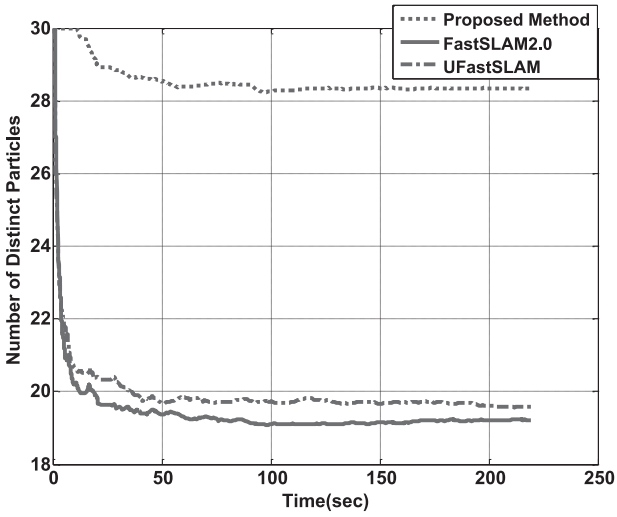


Figure 6. Number of distinct particles.

Table 3. Computational cost of algorithms.

Number of Particles	Processing time		
	Fast SLAM2.0	UFastSLAM	Proposed Method
30	43.4	78.8	66.5
20	34.1	55.6	49.9
10	25.3	36.7	31.4

4.1.3. Computational cost

The computational cost of the algorithms is analysed using the Matlab simulations on an Intel Core2Duo@3Ghz laptop. As seen in Table 3, the required computation time of the proposed method is lesser than that of UFastSLAM. The processing time of FastSLAM2.0 is lower than that of UFastSLAM and the proposed approach due to its

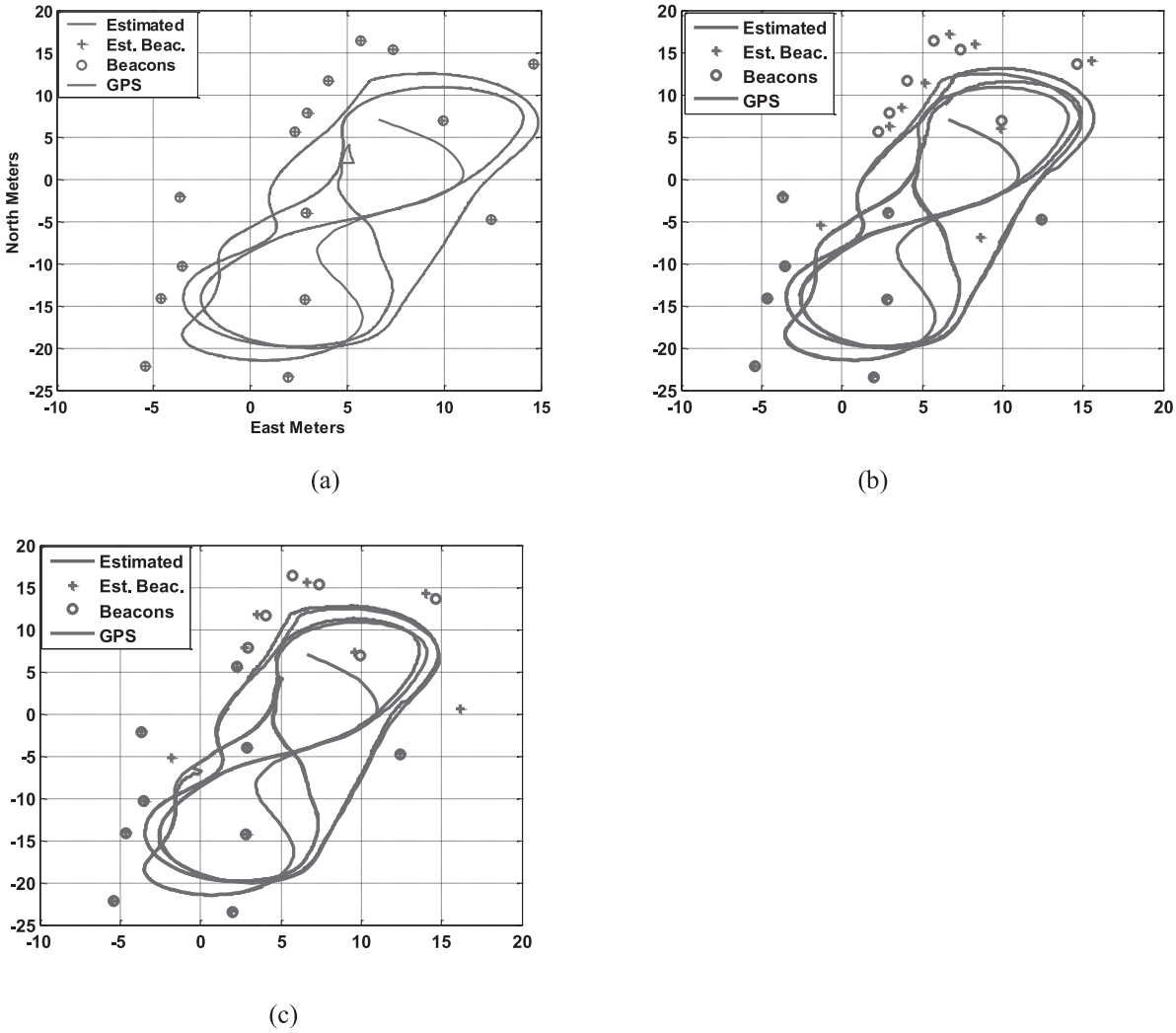


Figure 7. (a) Proposed method, (b) UFastSLAM and (c) FastSLAM2.0. The '. . .' is the path estimated, the ' + ' are the estimated beacon positions, the '__' is the GPS path reference and the 'o' are the beacon positions given by the GPS.

simplicity, but this fact is insignificant if its poor performance is taken into account.

4.2. Experimental results

The performance of the proposed method is evaluated on the Car Park data set and Victoria data set, two popular data sets in the SLAM community. These two data sets are available in Nebot. The experimental platform is a four-wheeled vehicle equipped with wheel encoders, GPS and a laser sensor. The vehicle had a 2.83 m wheelbase and was equipped with the SICK laser range finder with 180° frontal field of view.

In Car Park test, the vehicle was driven around the park. The artificial landmarks were used that consisted of 60 mm steel poles covered with reflective tape. With this approach, the feature extraction becomes trivial and the landmark observation model will be very accurate. Since the true position of the landmarks was also obtained with GPS, a true navigation map was available for comparison purposes. Furthermore, a GPS receiver is used to provide ground truth for the robot position. The performance of algorithms is compared in situation that the correspondence between the observation and the landmarks is assumed to be unknown and 20 particles are used for all algorithms. Each algorithm is run many times to confirm the variance of the estimate error. For the unknown data association, the individual compatibility in nearest neighbour test with 2σ acceptance region is used. Figure 7 shows the trajectory and landmark estimates produced by algorithms, while Figure 8 shows RMSE of the robot position with 20 particles and Figure 9 shows the performance of algorithms with various numbers of particles. The results show that the performance of the proposed algorithm is better than that of other algorithms.

The second experimental run is done in Victoria data set that is a larger area with mild uneven terrain and different types of surfaces. The vehicle was driven in the park around for about 30 min and was moved over 4 km, with a sensor to measure the velocity and the steering angle. Figure 10 shows Victoria Park with the GPS path. The GPS data were collected to provide ground truth data. Although the vehicle was equipped with the GPS, due to occlusion by foliage and buildings, ground truth data were not available over the whole experiment. However, the ground true position of the vehicle from the GPS was good enough to validate the estimated vehicle state. Figure 11 presents the trajectories and map for each algorithm. The results show that the performance of the proposed method is better than that of other algorithms. This is because the estimated trajectory of the proposed method coincides very well with the GPS paths.

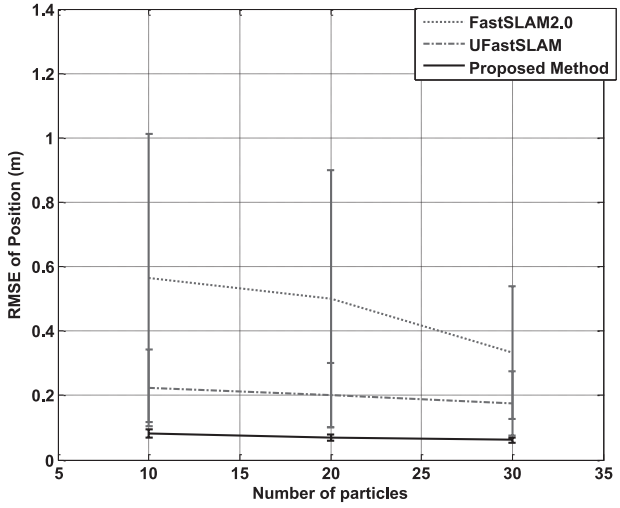


Figure 9. RMSE of algorithms with different numbers of particles.

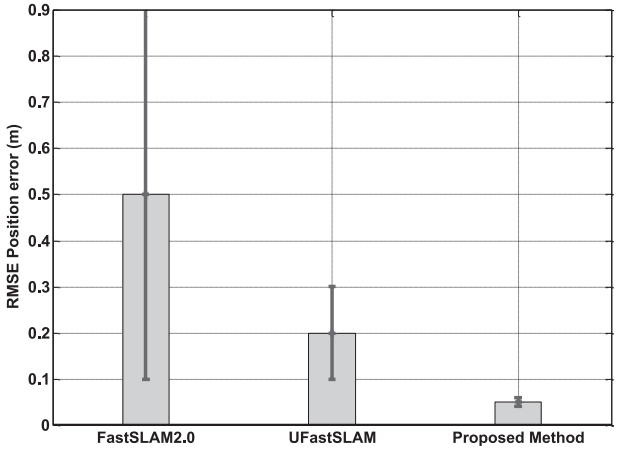


Figure 8. RMSE of the robot position.

Figure 10. Victoria Park with the GPS path.

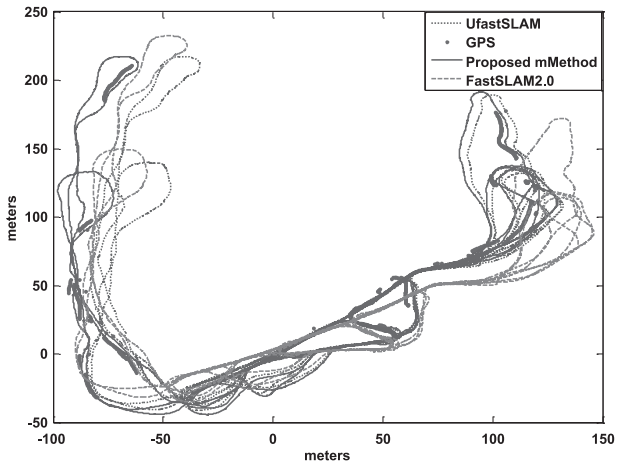


Figure 11. Comparison of algorithms.

5. Conclusion

FastSLAM is a framework for SLAM using a Rao-Blackwellised particle filter. However, this approach is inconsistent in long term. The structure of FastSLAM is one of the main reasons for inconsistency. In implementation of FastSLAM, resampling carried out over the marginal space. However, as the SLAM problem does not exhibit 'exponential forgetting' of its past errors, the resampling step provides an inconsistent estimation. To solve this problem, SLAM based on hybrid auxiliary marginalised particle filter and DE is proposed in this study. In this approach, samples are derived directly in the marginal space that avoids performing importance sampling on a space of growing dimension. The static map is also considered as a set of parameters that are learned using DE. In addition, the use of the IFGT to reduce the computational complexity of the proposed method has been considered. The performance of the proposed method is compared with UFastSLAM and FastSLAM2.0 for benchmark environment. The results show the effectiveness of the proposed method. The main advantage of the proposed approach is that the consistency and accuracy of the state estimation is improved in comparison with the other algorithms.

Notes on contributors

Ramazan Havangi received the M.Sc. degree in control engineering from the K. N. Toosi University of Technology, Tehran, Iran, in 2003, where he is currently working towards the Ph.D. degree in control engineering. His current research interests include inertial navigation, integrated navigation, estimation and filtering, evolutionary filtering, simultaneous localisation 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 from 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.T. 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 optimisation, linear systems, optimal control and different aspects of mathematics in control.

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 co-author 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.

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. degree in mechanical engineering in 1993, and his Ph.D. degree 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 organising committee member of many international conferences in the field of robotics, including International Conference on Robotics and Mechatronics (ICRoM). His publications include 5 books, and more than 190 papers in peer-reviewed international journals and conference proceedings.

References

Bailey, T., Nieto, J., and Nebot, E. (2006), 'Consistency of the FastSLAM Algorithm', in *Proceedings of the IEEE International Conference, on Robotics and Automation*, pp. 424–429.

Bar-Shalom, Y., Li, X.R., and Kirubarajan, T. (2001), *Estimation With Applications to Tracking and Navigation*, New York, NY: John Wiley and Sons.

Chang, C.S., Lu, L.R., and Wang, F. (2004), 'Application of Differential Evolution in Harmonic Worst-Case Identification of Mass Rapid Transit Power Supply System', *International Journal of Systems Science*, 35(13 & 14), 731–739.

Chen, T., Morris, J., and Martin, E. (2005), 'Particle Filters for State and Parameter Estimation in Batch Processes', *Journal of Process Control*, 665–673.

Crisan, D., and Doucet, A. (2002), 'A Survey of Convergence Results on Particle Filtering Methods for Practitioners', *IEEE Transactions on Signal Process*, 736–746.

Dissanayake, M.W.M.G., Newman, P., Clark, S., and Durrant-Whyte, H.F. (2001), 'A Solution to the Simultaneous Localization and Map Building (SLAM) Problem', *IEEE Transactions on Robotics and Automation*, 17(3), 229–241.

Doucet, A., and Tadic, V.B. (2003), 'Parameter Estimation in General State-Space Models Using Particle Methods', *Annals of the Institute of Statistical Mathematics*, 409–422.

Elgammal, A., Duraiswami, R., and Davis, L.S. (2003), 'Efficient Kernel Density Estimation Using the Fast Gauss Transform With Applications to Color Modeling and Tracking', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11), 1499–1504.

Engelbrecht, A.P. (2007), *Computational Intelligence an Introduction* (2nd ed.), New York, NY: John Wiley & Sons.

Greengard, L., and Strain, J. (1991), 'The Fast Gauss Transform', *SIAM Journal on Scientific Computing*, 12(1), 79–94.

Available online: http://www.personal.acfr.usyd.edu.au/tbailey/software/index.html.

Hua, W., and Shi-Yin, Q. (2011), 'An Approach to Robot SLAM Based on Incremental Appearance Learning With Omnidirectional Vision', *International Journal of Systems Science*, 42(3), 407–427.

Huang, S., and Dissanayake, G. (2007), 'Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM', *IEEE Transactions on Robotics*, 23(5), 1036–1049.

Kantas, N., Doucet, A., Singh, S.S., and Maciejowski, J. (2009), 'An Overview of Sequential Monte Carlo Methods for Parameter Estimation on General State Space Models', in *Proceedings of the 15th IFAC Symposium on System Identification*, Vol. 15, pp. 774–785.

Kim, I., Kwak, N., Lee, H., and Lee, B. (2009), 'Improved Particle Fusing Geometric relation Between Particles in FastSLAM', *Journal of Robotica*, 27, 853–859.

Kim, C., Sakthivel, R., and Chung, W.K. (2007), 'Unscented FastSLAM: A Robust Algorithm for the Simultaneous Localization and Mapping Problem', in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2439–2445.

Kim, C., Sakthivel, R., and Chung, W.K. (2008), 'Unscented FastSLAM: A Robust and Efficient Solution to the SLAM Problem', *IEEE Transactions on Robotics*, 24(4), 808–819.

Klaas, M., de Freitas, N., and Doucet, A. (2005), 'Toward Practical N2 Monte Carlo: The Marginal Particle filter', in *Proceedings of the 21st Conference Uncertainty Artificial Intelligence*, Edinburgh, Scotland, pp. 308–315.

Kwak, N., Kim, G., and Lee, B. (2007a), 'A New Compensation Technique Based on Analysis of Resampling Process in FastSLAM', *Journal of Robotica*, 26, 205–217.

Kwak, N., Kim, I.K., Lee, H.C., and Lee, B.H. (2007b), 'Adaptive Prior Boosting Technique for the Efficeint Sample Size in FastSLAM', in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 630–635.

Lee, H., Park, S., Choi, J., and Lee, B. (2009), 'PSO-FastSLAM: An Improved FastSLAM Framework Using Particle Swarm Optimization', in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2763–2768.

Martinez-Cantin, R., de Freitas, N., and Castellanos, J.A. (2007), 'Analysis of Particle Methods for Simultaneous Robot Localization and Mapping and a New Algorithm: Marginal-SLAM', in *IEEE Intlernational Conference on Robotics and Automation*, pp. 2415–2420.

Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. (2003), 'Fastslam 2.0: An Improved Particle filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges', in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1151–1156.

Nebot, E. Victoria park data set. http://www-personal.acfr.usyd.edu.au/nebot/dataset.htm.

Poyiadjis, G., Doucet, A., and Singh, S.S. (2005), 'Maximum Likelihood Parameter Estimation in General State-Space Models Using Particle Methods', in *Proceedings of the American Statistical Association*, Minneapolis, pp. 2279–2289.

Rodriguez-Losada, D., San Segundo, P., Matia, F., and Pedraza, L. (2009), 'Dual FastSLAM: Dual Factorization of the Particle Filter Based Solution of the Simultaneous Localization and Mapping Problem', *Journal of Intelligent and Robot Systems*, 109–134.

Thrun, S., Montemerlo, M., Koller, D., Wegbreit, B., Nieto, J., and Nebot, E. (2004), 'FastSLAM: An Efficient Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association', *Journal of Machine Learning Research*, 380–407.

Wills, A.G., Schon, T., and Ninness, B. (2008), 'Parameter Estimation for Discrete-Time Nonlinear Systems Using EM', in *Proceedings of the 17th IFAC World Congress*, Seoul, South Korea, pp. 4012-4017.

Yang, C., Duraiswami, R., Gumerov, N.A., and Davis, L. (2003), 'Improved Fast Gauss Transform and Efficient Kernel Density Estimation', in *Proceedings of IEEE International Conference on Computer Vision*, pp. 464–471.

Appendix. Reducing the computational complexity

For computing the proposal distribution and importance weight, each particle interacts with a density represented by N particles as follows:

$$q_1^{[i]} = \sum_{j=1}^{N} w_{t-1}^{[j]} p\left(s_t^{[i]} | s_{t-1}^{[j]}\right). \tag{A1}$$

In general, this equation can be reformulated as the weighted fast kernel density estimation (KDE) problem as follows:

$$q^{[i]} = \sum_{j=1}^{N} \omega^{[j]} K(s^{[j]}, t^{[i]}), \tag{A2}$$

where

$$K(s^{[j]}, t^{[i]}) = p\left(s_t^{[i]} = t^{[i]} | s_{t-1}^{[j]} = s^{[j]}\right). \tag{A3}$$

Also $s^{[j]}\mathbb{R}^D$ is the source points with associated weights $w^{[j]}$, $t^{[i]}\mathbb{R}^D$ is the target points and $K(s^{[j]}, t^{[i]})$ is the kernel function. There have been several approaches suggested in the literature to reduce this computational complexity while compromising on the accuracy. Different methods for fast evaluation of KDE rely

on the division approach where either the source space or the joint source and target space are first partitioned into different regions. In our problem, the kernel K is Gaussian as

$$K = \frac{1}{\sqrt{(2\pi)^n\,|P_t|}} e^{-\frac{1}{2}(s_t^{[i]}-m_t^{[j]})^T P_t^{-1}(s_t^{[i]}-m_t^{[j]})}. \tag{A4}$$

In which, $m_t^{[j]}$ and P_t are mean and covariance of probability Gaussian (kernel), respectively. By Cholesky factorisation, positive-definite matrix P_t may be decomposed into the product of an upper triangular matrix and its transpose as

$$P_t = R_t^T R_t. \tag{A5}$$

Using this factorisation, the kernel K can rewrite as follows:

$$K = \frac{1}{\sqrt{(2\pi)^n\,|P_t|}} e^{-\frac{1}{2}\left\|\tau_t^{[i]}-\delta_t^{[j]}\right\|^2}, \tag{A6}$$

where $\tau_t^{[i]} = (R_t^T)^{-1} s_t^{[i]}$ and $\delta_t^{[j]} = (R_t^T)^{-1} m_t^{[j]}$. Therefore, the kernel K can be formulated as the weighted KDE problem as follows:

$$\begin{cases} q^{[i]} = \sum_{j=1}^{N} \varpi_t^{[j]} \exp\left(- \left\|\tau_t^{[i]} - \delta_t^{[j]}\right\|^2 \Big/ \sigma^2 \right) \\ \varpi_t^{[j]} = \dfrac{\omega_j}{\sqrt{(2\pi)^n\,|P_t|}} \quad \sigma = \sqrt{2} \end{cases}, \tag{A7}$$

where Equation (A7) is adapted to the standard form of KDE with Gaussian kernel. Therefore, the fast Gaussian transform (FGT) (Greengard and Strain 1991; Elgammal, Duraiswami, and Davis 2003) and IFGT (Yang, Duraiswami, Gumerov, and Davis 2003) can be used to reduce the computational complexity of KDE in $O(N)$. In this paper, the use of IFGT to reduce the computational complexity of KDE has been considered. This is because computational constant of IFGT grows more moderately with the dimension compared to FGT, which the computational constant grows exponentially with the dimension. Details about IFGT can be found in Elgammal et al. (2003) and Yang et al. (2003).