Robotica (2009) volume 27, pp. 411–423. © 2008 Cambridge University Press
doi:10.1017/S0263574708004827 Printed in the United Kingdom

Differential evolution tuned fuzzy supervisor adapted extended Kalman filtering for SLAM problems in mobile robots

Amitava Chatterjee

Jadavpur University, Electrical Engineering Department, Kolkata, West Bengal 700032, India
E-mail: *achatterjee@ee.jdvu.ac.in*

(Received in Final Form: May 30, 2008. First published online: July 4, 2008)

SUMMARY

The present paper proposes a successful application of differential evolution (DE) optimized fuzzy logic supervisors (FLS) to improve the quality of solutions that extended Kalman filters (EKFs) can offer to solve simultaneous localization and mapping (SLAM) problems for mobile robots and autonomous vehicles. The utility of the proposed system can be readily appreciated in those situations where an incorrect knowledge of **Q** and **R** matrices of EKF can significantly degrade the SLAM performance. A fuzzy supervisor has been implemented to adapt the **R** matrix of the EKF online, in order to improve its performance. The free parameters of the fuzzy supervisor are suitably optimized by employing the DE algorithm, a comparatively recent method, popularly employed now-a-days for high-dimensional parallel direct search problems. The utility of the proposed system is aptly demonstrated by solving the SLAM problem for a mobile robot with several landmarks and with wrong knowledge of sensor statistics. The system could successfully demonstrate enhanced performance in comparison with usual EKF-based solutions for identical environment situations.

KEYWORDS: Extended Kalman filter; Fuzzy supervisor; Differential evolution; Sensor statistics; SLAM problem.

1. Introduction

The simultaneous localization and mapping (SLAM) problem has been regarded as a complex and important present-day research problem within the research community of mobile robots and autonomous vehicle. The problem under consideration is the navigation of a mobile platform in an environment where both the map of the environment and the localization of the mobile platform are unknown. The problem is solved iteratively where both localization of the robot/vehicle and determination of the environment map are estimated simultaneously.[1,2] Some successful solutions for this complicated problem have so far been proposed employing more traditional extended Kalman filter (EKF)[3–6] and more recent particle filtering[2] based algorithms. In EKF-based solution, the problem is solved by maintaining a complete state vector of the robot pose and the position of each landmark observed in the environment. The uncertainties associated with this estimation process are kept stored in covariance matrices. The solution of such a problem by configuring it as a state estimation problem is more complicated than conventional state estimation problems because the state vector is dynamic in size and the number of landmarks observed and which landmarks are observed vary with time instants. Furthermore, there is always the possibility that both the vehicle's pose estimate and its associated map estimates become increasingly inaccurate in absence of any global position information.[2]

The performance of an EKF-based SLAM problem will depend on how accurately the statistics depicting measurement uncertainties and process uncertainties are known. EKF assumes independent Gaussian-distribution-based measurement noise and process noise models with known covariance and these are specified in form of process covariance matrix (**Q**) and measurement noise covariance matrix (**R**). An incorrect *a priori* knowledge of **Q** and **R** (which is quite possible in real-world scenario) may lead to performance degradation[7] and it can even lead to practical divergence.[8] In fact, for SLAM problems our experiments have shown that the performance can become significantly worse under these situations. In this scheme, a fuzzy-logic-based system is proposed, which will adapt the **R** matrix of EKF-based SLAM problem online, so that the solution offered by the EKF can be kept satisfactory.

Although fuzzy logic has been employed previously to tune traditional Kalman filters with fixed number of states by adapting both **Q** and **R** simultaneously or adapting **R** only,[9,10] our algorithm considers a much more complicated problem in SLAM domain where the sizes of the state vector and hence the covariance matrix keep varying with iterations. Hence it is very important that the free parameters of the fuzzy system are tuned properly so that the fuzzy adaptation, online, can provide meaningful supervision of the performance of EKF for SLAM problems. The present paper proposes the implementation of a contemporary parallel direct search method, popularly employed for stochastic global optimization problems, called differential evolution (DE). In this problem, DE has been employed to learn the free parameters of the fuzzy system. DE has recently evolved as a promising methodology that can be used for minimizing real-valued, high-dimensional, multimodal objective functions.[11] The main strength of DE lies in the fact that it is quite simple to use and, yet, capable of providing satisfactory solutions.[12] Such types of stochastic, global optimization methods are especially suitable for the problem under our consideration, because they are non-gradient-based optimization techniques and it is very difficult to employ gradient-based techniques for learning the fuzzy supervisor that is under our consideration. However, in this context, it should be pointed out that now there are several such candidate-population-based, non-gradient-type, stochastic, global optimization methods available in literature, e.g. genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and bacterial foraging, and each one can be potentially applied for solving such a problem. In this literature, the performance of DE is compared with that of PSO, in tuning the fuzzy supervisor for adapting the EKF-based SLAM algorithm, and the usefulness of the DE algorithm in this regard is demonstrated.

The rest of the paper is organized as follows. Section 2 presents the fuzzy supervisor aided adaptive extended Kalman filtering algorithm for SLAM problems. Section 3 presents a short primer on the DE strategy that has been successfully employed to learn the fuzzy supervisor. Section 4 presents the performance evaluation. Conclusions are presented in Section 5.

2. Adaptive EKF-Based SLAM Algorithm Employing Fuzzy Supervisor

Detailed discussions on conventional EKF-based SLAM algorithms are presented in ref. [4]. In EKF, the state transition can be modeled by a nonlinear function $\mathbf{f}(\bullet)$ and the observation or measurement of the state can be modeled by a nonlinear function $\mathbf{h}(\bullet)$, given as

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{q}_k, \tag{1a}$$

$$\mathbf{z}_{k+1} = \mathbf{h}(\mathbf{x}_{k+1}) + \mathbf{r}_{k+1}. \tag{1b}$$

Let the robot/vehicle states be represented by their pose with a mean estimate $\hat{x}_v$ and the covariance matrix $\mathbf{P}_v$, defined as

$$\hat{\mathbf{x}}_v = [\hat{x}_v \quad \hat{y}_v \quad \hat{\varphi}_v]^T, \tag{2a}$$

$$\mathbf{P}_v = \begin{bmatrix} \sigma^2_{x_v x_v} & \sigma^2_{x_v y_v} & \sigma^2_{x_v \varphi_v} \\ \sigma^2_{x_v y_v} & \sigma^2_{y_v y_v} & \sigma^2_{y_v \varphi_v} \\ \sigma^2_{x_v \varphi_v} & \sigma^2_{y_v \varphi_v} & \sigma^2_{\varphi_v \varphi_v} \end{bmatrix}. \tag{2b}$$

Considering that there are n 2D static, point features observed in the map, the position estimates of these features are given by their mean estimate $\hat{\mathbf{x}}_m$ and the covariance matrix $\mathbf{P}_m$, given as

$$\hat{\mathbf{x}}_m = [\hat{x}_1 \quad \hat{y}_1 \quad \dots \quad \hat{x}_n \quad \hat{y}_n]^T, \tag{3}$$

$$\mathbf{P}_m = \begin{bmatrix} \sigma^2_{x_1 x_1} & \sigma^2_{x_1 y_1} & \cdots & \sigma^2_{x_1 x_n} & \sigma^2_{x_1 y_n} \\ \sigma^2_{x_1 y_1} & \sigma^2_{y_1 y_1} & \cdots & \sigma^2_{y_1 x_n} & \sigma^2_{y_1 y_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma^2_{x_1 x_n} & \sigma^2_{y_1 x_n} & \cdots & \sigma^2_{x_n x_n} & \sigma^2_{x_n y_n} \\ \sigma^2_{x_1 y_n} & \sigma^2_{y_1 y_n} & \cdots & \sigma^2_{x_n y_n} & \sigma^2_{y_n y_n} \end{bmatrix}. \tag{4}$$

In SLAM, a total state vector of the estimates of vehicle pose and feature positions is kept in the form of the mean vector $\hat{x}_a$ and the corresponding correlation matrix $\mathbf{P}_a$. These are given as

$$\hat{\mathbf{x}}_a = \hat{\mathbf{x}} = \left[\hat{\mathbf{x}}_v^T \quad \hat{\mathbf{x}}_m^T\right]^T, \tag{5}$$

$$\mathbf{P}_a = \mathbf{P} = \begin{bmatrix} \mathbf{P}_v & \mathbf{P}_{vm} \\ \mathbf{P}_{vm}^T & \mathbf{P}_m \end{bmatrix}, \tag{6}$$

where $\mathbf{P}_{vm}$ maintains the robot–map correlation. The system is initialized assuming that $\hat{\mathbf{x}}_a = \hat{\mathbf{x}}_v = 0$ and $\mathbf{P}_a = \mathbf{P}_v = 0$.

2.1. Prediction step

In each prediction step, the augmented state vector estimate $\hat{\mathbf{x}}_a^-$ in the next sampling instant is predicted on the basis of the augmented state vector at the present sampling instant $\hat{\mathbf{x}}_a$ and the change in vehicle pose between successive sampling instants (separated by the discrete time step Δt). The control input vector $\mathbf{u} = [v \ s]^T$ is composed of the steering angle command (s) and the velocity at which the rear wheel of the robot is driven (v). So the state estimates can be obtained by employing wheel encoder odometry and the robot kinematic model. However, the control inputs v and s must be considered with the uncertainties involved due to, e.g., wheel slippage and incorrect calibration of vehicle controller. These uncertainties are modeled as Gaussian variations in v and s from their nominal values. Hence, the prediction step calculates

$$\hat{\mathbf{x}}_a^- = \begin{bmatrix} \mathbf{f}_v(\hat{\mathbf{x}}_v, \mathbf{u}) \\ \hat{\mathbf{x}}_m \end{bmatrix}, \tag{7}$$

$$\mathbf{P}_a^- = \begin{bmatrix} \nabla\mathbf{g}_{\mathbf{x}_v}\mathbf{P}_v\nabla\mathbf{g}_{\mathbf{x}_v}^T + \mathbf{Q} & \nabla\mathbf{g}_{\mathbf{x}_v}\mathbf{P}_{vm} \\ (\nabla\mathbf{g}_{\mathbf{x}_v}\mathbf{P}_{vm})^T & \mathbf{P}_m \end{bmatrix}, \tag{8}$$

where $\mathbf{f}_v$ estimates the robot pose on the basis of the motion model and the control inputs and $\mathbf{Q}$ gives the covariance matrix for $\mathbf{f}_v$. $\mathbf{Q}$ is calculated as

$$\mathbf{Q} = \nabla\mathbf{g}_{\mathbf{u}} U \nabla\mathbf{g}_{\mathbf{u}}^T. \tag{9}$$

The Jacobians and $\mathbf{U}$, the covariance matrix of $\mathbf{u}$, are given as

$$\nabla\mathbf{g}_{\mathbf{x}_v} = \left.\frac{\partial\mathbf{f}_v}{\partial\mathbf{x}_v}\right|_{(\hat{\mathbf{x}}_v,\mathbf{u})}, \quad \nabla\mathbf{g}_{\mathbf{u}} = \left.\frac{\partial\mathbf{f}_v}{\partial\mathbf{u}}\right|_{(\hat{\mathbf{x}}_v,\mathbf{u})}, \quad \text{and}$$

$$\mathbf{U} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_s^2 \end{bmatrix}. \tag{10}$$

Because the features are assumed to remain stationary with time, $\hat{\mathbf{x}}_m$ and $\mathbf{P}_m$ in the previous equations remain constant with time.

2.2. Update step

Let us assume that an existing feature is reobserved i.e. the ith feature $(\hat{x}_i, \hat{y}_i)$. Let this feature be measured in terms of its range (r) and bearing (θ) relative to the observer, given as

$$\mathbf{z} = [r \quad \theta]^T. \tag{11}$$

The uncertainties in these observations are again modeled by Gaussian variations and let **R** be the corresponding observation/measurement noise covariance matrix given as

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}, \tag{12}$$

where it is assumed that there is no cross-correlation between the range and bearing measurements. In the context of the map, the measurements can be given as

$$\hat{\mathbf{z}}_i = \mathbf{h}_i(\hat{\mathbf{x}}_a) = \begin{bmatrix} \sqrt{(\hat{x}_i - \hat{x}_v)^2 + (\hat{y}_i - \hat{y}_v)^2} \\ \arctan\left(\dfrac{\hat{y}_i - \hat{y}_v}{\hat{x}_i - \hat{x}_v}\right) - \hat{\varphi}_v \end{bmatrix}. \tag{13}$$

Now the Kalman gain $\mathbf{W}_i$ can be calculated assuming that there is correct landmark association between **z** and $(\hat{x}_i, \hat{y}_i)$ and the following computations can be resorted to

$$v_i = \mathbf{z} - \mathbf{h}_i(\hat{\mathbf{x}}_a^-), \tag{14}$$

$$\mathbf{S}_i = \nabla\mathbf{h}_{\mathbf{x}_a}\mathbf{P}_a^- \nabla\mathbf{h}_{\mathbf{x}_a}^T + \mathbf{R}, \tag{15}$$

$$\mathbf{W}_i = \mathbf{P}_a^- \nabla\mathbf{h}_{\mathbf{x}_a}^T \mathbf{S}_i^{-1}, \tag{16}$$

where v_i denotes the innovation of the observation for this ith landmark and $\mathbf{S}_i$ the associated innovation covariance matrix. The Jacobian $\nabla\mathbf{h}_{\mathbf{x}_a}$ is given as:

$$\nabla\mathbf{h}_{\mathbf{x}_a} = \left.\frac{\partial\mathbf{h}_i}{\partial\mathbf{x}_a}\right|_{\hat{\mathbf{x}}_a^-}. \tag{17}$$

Hence, the *a posterior* augmented state estimate and the corresponding covariance matrix are updated as

$$\hat{\mathbf{x}}_a^+ = \hat{\mathbf{x}}_a^- + \mathbf{W}_i v_i, \tag{18}$$

$$\mathbf{P}_a^+ = \mathbf{P}_a^- - \mathbf{W}_i\mathbf{S}_i\mathbf{W}_i^T. \tag{19}$$

However, during the execution of SLAM algorithm, it is highly likely that a new feature is observed for the first time. Then this should be initialized into the system by incorporating their 2D position coordinates in the augmented state vector and accordingly modifying the covariance matrix. These new $\hat{\mathbf{x}}_a^+$ and $\mathbf{P}_a^+$ can be calculated as

$$\hat{\mathbf{x}}_a^+ = \begin{bmatrix} \hat{\mathbf{x}}_a \\ \mathbf{f}_i(\hat{\mathbf{x}}_v, \mathbf{z}) \end{bmatrix}, \tag{20}$$

$$\mathbf{P}_a^+ = \begin{bmatrix} \mathbf{P}_v & \mathbf{P}_{vm} & \mathbf{P}_v\nabla\mathbf{g}_{\mathbf{x}_v}^T \\ \mathbf{P}_{vm}^T & \mathbf{P}_m & \mathbf{P}_{vm}^T\nabla\mathbf{g}_{\mathbf{x}_v}^T \\ \nabla\mathbf{g}_{\mathbf{x}_v}\mathbf{P}_v & \nabla\mathbf{g}_{\mathbf{x}_v}\mathbf{P}_{vm} & \nabla\mathbf{g}_{\mathbf{x}_v}\mathbf{P}_v\nabla\mathbf{g}_{\mathbf{x}_v}^T + \nabla\mathbf{g}_{\mathbf{z}}\mathbf{R}\nabla\mathbf{g}_{\mathbf{z}}^T \end{bmatrix}. \tag{21}$$

Here $\mathbf{f}_i(\hat{\mathbf{x}}_v, \mathbf{z})$ is employed to convert the polar observation **z** to the base Cartesian coordinate frame. The Jacobians are calculated as

$$\nabla\mathbf{g}_{\mathbf{x}_v} = \left.\frac{\partial\mathbf{f}_i}{\partial\mathbf{x}_v}\right|_{(\hat{\mathbf{x}}_v,\mathbf{z})} \quad \text{and} \quad \nabla\mathbf{g}_{\mathbf{z}} = \left.\frac{\partial\mathbf{f}_i}{\partial\mathbf{z}}\right|_{(\hat{\mathbf{x}}_v,\mathbf{z})}. \tag{22}$$

Similarly, any unreliable feature can be deleted by deleting the relevant row entry from the state vector and the relevant row and column entries from the covariance matrix.

Now, a fuzzy supervision scheme is proposed to adapt the **R** matrix, online, to improve the performance of the EKF. The concept utilized is similar to the fuzzy-logic-based innovation adaptive estimation (IAE) approach,[10] where new statistical information from innovation sequence is utilized to correct the estimation of the states. The basic concept relies on determining the discrepancy between a new measurement $\mathbf{z}_k$ and its corresponding predicted estimation $\hat{\mathbf{z}}_k$, at any arbitrary kth instant, and utilizing this new information to correct the estimations/predictions already made. In our SLAM algorithm, actual covariance of the innovation sequences, $\mathbf{C}_{\mathrm{Inn}k}$, is calculated as

$$\mathbf{C}_{\mathrm{Inn}k} = v_k\, v_k^T, \tag{23}$$

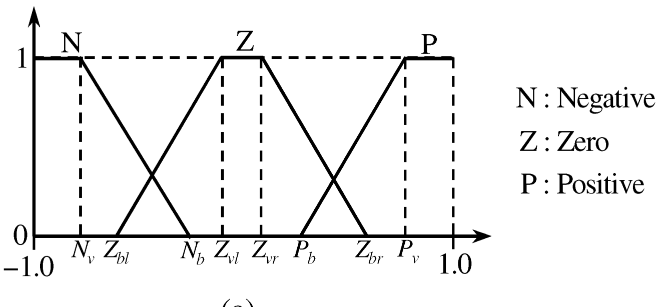
where v_k denotes the augmented innovation sequence, made consistent with the batch mode of observations. The objective is to minimize the mismatch between $\mathbf{C}_{\mathrm{Inn}k}$ and the theoretical covariance of the innovation sequences ($\mathbf{S}_k$), made consistent with the batch mode of observations. The mismatch, at the kth instant, is given as

$$\Delta\mathbf{C}_{\mathrm{Inn}k} = \mathbf{C}_{\mathrm{Inn}k} - \mathbf{S}_k. \tag{24}$$

Compared to conventional EKF algorithms, the EKF-based SLAM is much more complex, because the size of v_k, $\mathbf{C}_{\mathrm{Inn}k}$, and $\mathbf{S}_k$ keep changing from iteration to iteration because each of them is dependent on the number of features observed during any given *observation and update step*, which were all observed at least once, before.

Now, a fuzzy system is employed to minimize the mismatch given in (24). The overall fuzzy system employs a bank of subsystems where each subsystem employs a SISO fuzzy system. These single-input-single-output (SISO) fuzzy systems are utilized to use each diagonal element of the $\Delta\mathbf{C}_{\mathrm{Inn}k}$ matrix, i.e. $\Delta\mathbf{C}_{\mathrm{Inn}k}(j, j)$, as the input, and the output is $\Delta\mathbf{R}(j, j)$, an adaptation recommended for the corresponding diagonal element of the augmented measurement noise covariance matrix **R**, computed according to the batch-mode situation. The size of the augmented **R** matrix varies with iteration and it is $[2z_f \times 2z_f]$ where z_f is the number of landmarks observed in that iteration, which were also observed earlier. This augmented **R** is formed utilizing the original $[2 \times 2]$ **R** matrix and this is formulated as

$$\text{augmented } \mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_\theta^2 & \cdots & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \sigma_r^2 & 0 & \cdots & \vdots & \vdots \\ \vdots & \vdots & 0 & \sigma_\theta^2 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & \sigma_r^2 & 0 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & \sigma_\theta^2 \end{bmatrix}. \tag{25}$$



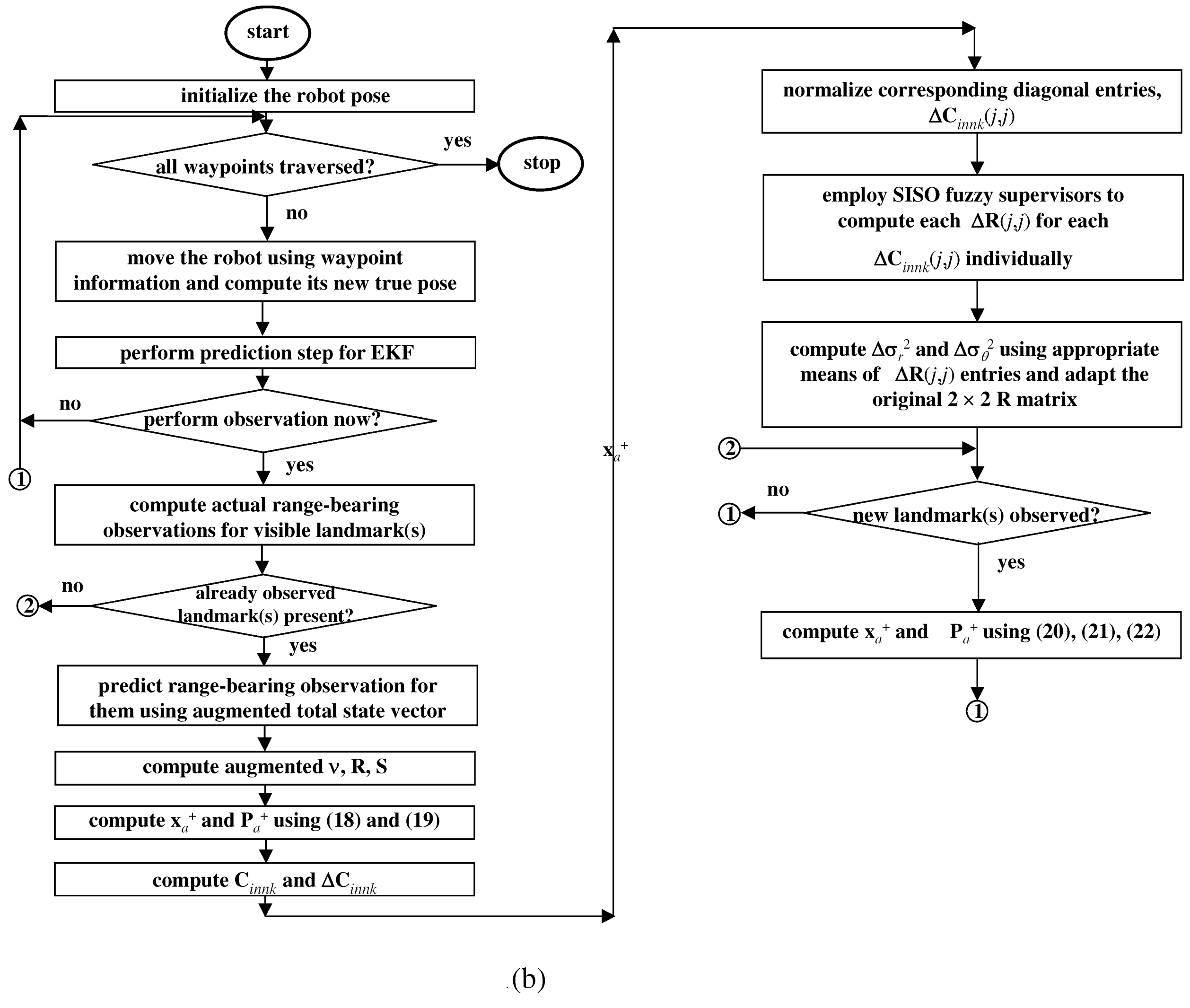

start
initialize the robot pose
all waypoints traversed?
yes
stop
no
move the robot using waypoint information and compute its new true pose
perform prediction step for EKF
no
perform observation now?
yes
①
compute actual range-bearing observations for visible landmark(s)
no
②
already observed landmark(s) present?
yes
predict range-bearing observation for them using augmented total state vector
compute augmented ν, R, S
compute x_a^+ and P_a^+ using (18) and (19)
compute C_{innk} and ΔC_{innk}
x_a^+
normalize corresponding diagonal entries, $\Delta C_{innk}(j,j)$
employ SISO fuzzy supervisors to compute each $\Delta R(j,j)$ for each $\Delta C_{innk}(j,j)$ individually
compute $\Delta\sigma_r^2$ and $\Delta\sigma_\theta^2$ using appropriate means of $\Delta R(j,j)$ entries and adapt the original 2 × 2 R matrix
②
no
①
new landmark(s) observed?
yes
compute x_a^+ and P_a^+ using (20), (21), (22)
①
(b)

Fig. 1. (a) Membership functions chosen for each fuzzy subsystem. (b) Fuzzy supervisor adapted EKF-based SLAM algorithm, in implementation phase.

Here, σ_r^2 and σ_θ^2 correspond to the sensor statistics computed for that iteration. The same fuzzy system was employed for each of those SISO fuzzy subsystems to simplify the design, and hence normalized input for each fuzzy subsystem was employed. However, the ranges of mismatches in range and bearing observations may be of different order depending on the degree of accuracy/inaccuracy in the available knowledge of those specific sensor statistics. Hence $\Delta\mathbf{C}_{\text{Inn}k}(j,j)$ elements were segregated corresponding to range and bearing observations in two different sets and normalization of each fuzzy system input was employed corresponding to range and bearing observations separately. The input to each fuzzy subsystem is fuzzified employing three membership functions (MFs): negative (N), zero (Z), and positive (P), as shown in Fig. 1(a). Then each system employs three fuzzy rules

$$\text{if } \Delta\mathbf{C}_{\text{Inn}k}(j,j) \text{ is N then } \Delta\mathbf{R}(j,j) = w_1,$$
$$\text{if } \Delta\mathbf{C}_{\text{Inn}k}(j,j) \text{ is Z then } \Delta\mathbf{R}(j,j) = w_2, \text{ and}$$
$$\text{if } \Delta\mathbf{C}_{\text{Inn}k}(j,j) \text{ is P then } \Delta\mathbf{R}(j,j) = w_3.$$

The final output of each fuzzy system i.e. $\Delta\mathbf{R}(j, j)$ is calculated by employing the well-known weighted average technique. Finally the adaptations, i.e. $\Delta\sigma_r^2$ and $\Delta\sigma_\theta^2$, required for the original [2×2] **R** matrix, are computed on the basis of the appropriate means, separately computed from the arrays of $\Delta\mathbf{R}(j, j)$ entries for range and bearing observations. This adapted original [2×2] **R** matrix is employed to prepare the augmented R in (25) and it is utilized in the next *observation and update step* for the EKF-based SLAM algorithm. The complete algorithm with fuzzy supervisor, in the implementation phase, is given in Fig. 1(b).

Now, for satisfactory performance of the fuzzy adapted EKF, it is very important that the free parameters of the fuzzy system are properly chosen. These free parameters include the parameters of the three MFs and the three output gains of the three fuzzy rules for each SISO fuzzy subsystem. One of the prime difficulties encountered in training these parameters for the fuzzy systems is that the desired output for a given input for each of these fuzzy systems is not definitely known and hence we cannot employ usual supervised mode of neural network training to determine these parameters. Hence DE strategy was employed, a popular, contemporary technique in the category of direct search methods for stochastic global optimization problems. These methods can be very suitable in those situations where the optimization technique can be employed for minimization of a function without the knowledge of the goal i.e. the minimum value it is supposed to attain finally and they do not require any gradient information for their operation. Here, the free parameters of the fuzzy system are organized in the form of a vector $\mathbf{x} = (N_v\ N_b\ Z_{bl}\ Z_{vl}\ Z_{vr}\ Z_{br}\ P_b\ P_v\ w_1\ w_2\ w_3)$ and we attempt to minimize a cost function $f(\mathbf{x})$ on the basis of the $\mathbf{x}$. The cost function is designed as

$$f_f = \frac{\sum_{n_{\mathrm{obs}}=1}^{N_{\mathrm{obs}}} \left(\left(\sum_{j=1}^{J_{C\_n\mathrm{obs}}} [\Delta\mathbf{C}_{\mathrm{Inn}k}(j, j)]^2 \right) \Big/ J_{C\_\mathrm{obs}} \right)}{N_{\mathrm{obs}}}, \tag{26}$$

where N_{obs} denotes the total number of observation instants in a given iteration and $J_{C\_n\mathrm{obs}}$ denotes the total number of diagonal elements of $\Delta\mathbf{C}_{Innk}$ matrix when the n_{obs}th observation is made. While implementing this optimization strategy, the variables forming the high-dimensional vector $\mathbf{x}$ are implemented with several constraints to satisfy some shape constraints chosen for the MFs and also to ensure that there is some overlapping between the stretches of consecutive MFs. These constraints are given as

$$\begin{aligned}
&\text{if } (N_b < N_y) \quad \text{then } (N_b = N_v),\\
&\text{if } (Z_{vl} < Z_{bl}) \quad \text{then } (Z_{vl} = Z_{bl}),\\
&\text{if } (Z_{vr} < Z_{vl}) \quad \text{then } (Z_{vr} = Z_{vl}),\\
&\text{if } (Z_{br} < Z_{vr}) \quad \text{then } (Z_{br} = Z_{vr}),\\
&\text{if } (P_v < P_b) \quad \text{then } (P_v = P_b),\\
&\text{if } (N_b < Z_{bl}) \quad \text{then } (N_b = Z_{bl}), \text{ and}\\
&\text{if } (Z_{br} < P_b) \quad \text{then } (Z_{br} = P_b).
\end{aligned} \tag{27}$$

3. Differential Evolution-Based Optimization

In differential evolution, like many other population-based global optimization methods, several candidate solutions, each containing a possible solution vector for the optimization problem under consideration, are created simultaneously in the multidimensional search space and each one of them is individually evaluated in terms of its fitness function, which indicates the degree of suitability of that particular candidate solution to evolve as the best possible solution. This process is continued in an iterative fashion, where new vectors, i.e. possible candidate solutions, are created from the candidate solutions in the previous generation, in quest for generation of better and better solutions, which can be quantitatively evaluated by fitter and fitter fitness function values. Several mathematical strategies can be employed to create new candidate vectors for the current generation, based on the old candidate vectors of the previous generation. At the end of each generation, the candidate solution providing the fittest fitness function value (usually the minimum value) emerges as the best possible solution. This iterative process continues until the fittest fitness function value (usually the minimum value) for the best solution vector in a generation falls below the maximum permitted fitness function value for that optimization process or when the maximum number of generations is reached.

Let us consider that, in the basic variant of DE, utilized for minimizing a cost function $f(\mathbf{x})$ on the basis of D-dimensional $\mathbf{x}$, NP number of such candidate solutions of $(x_1, x_2, \ldots, x_D)$ are created in the D-dimensional space and the suitability of each of them is evaluated in each generation G. The initial population is generated in a random fashion and the objective is that the generated vectors should try to cover the entire search space as far as practicable. Each ith vector for the $(G+1)$th generation is created by adding the weighted difference between two population vectors to a third vector, all these three vectors pertaining to the Gth generation. This can be shown by the following formula[11,12]:

$$v_{i,G+1} = x_{r_1,G} + F(x_{r_2,G} - x_{r_3,G}), \tag{28}$$

where $i = 1, 2, \ldots, NP$. Here $r_1, r_2, r_3 \in [1, NP]$ and they are all mutually different. F is a constant weighting factor and usually $F \in [0, 2]$. This factor influences the amplification of the difference $(x_{r_2,G} - x_{r_3,G})$.

To increase diversity in the newly generated vector, the method of crossover is introduced. This crossover operation generates a new vector $\boldsymbol{u}_{i,G+1}$, from the newly generated perturbed vector $\boldsymbol{v}_{i,G+1}$ and the old vector $\boldsymbol{x}_{i,G}$. In the basic variant of DE, this new vector is generated as[11,12]

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}) \text{ with}$$
$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{for } j = \langle n+1 \rangle_D \cdots \langle n+L \rangle_D \\ x_{ji,G} & \text{for all other } j \in [1, D] \end{cases}. \tag{29}$$

Here, n is a randomly chosen integer, $n \in [1, D]$, and it determines the starting index for the crossover. The length or duration of crossover, in this basic variant of DE, is also

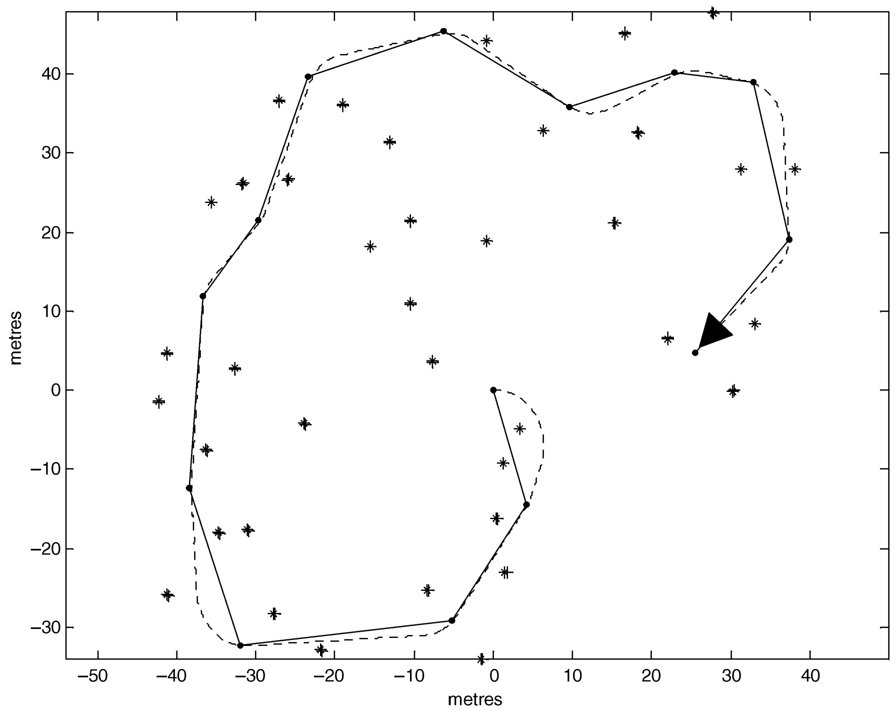


Fig. 2. Performance of the conventional EKF-based SLAM under correct knowledge of sensor statistics ($\sigma_r = 0.1$ m and $\sigma_b = 1$ deg.).

an integer drawn from the interval [1,D], and is based on the chosen crossover probability, $CR \in [0, 1]$. These n and L values are chosen afresh for each $\boldsymbol{u}_{i,G+1}$.

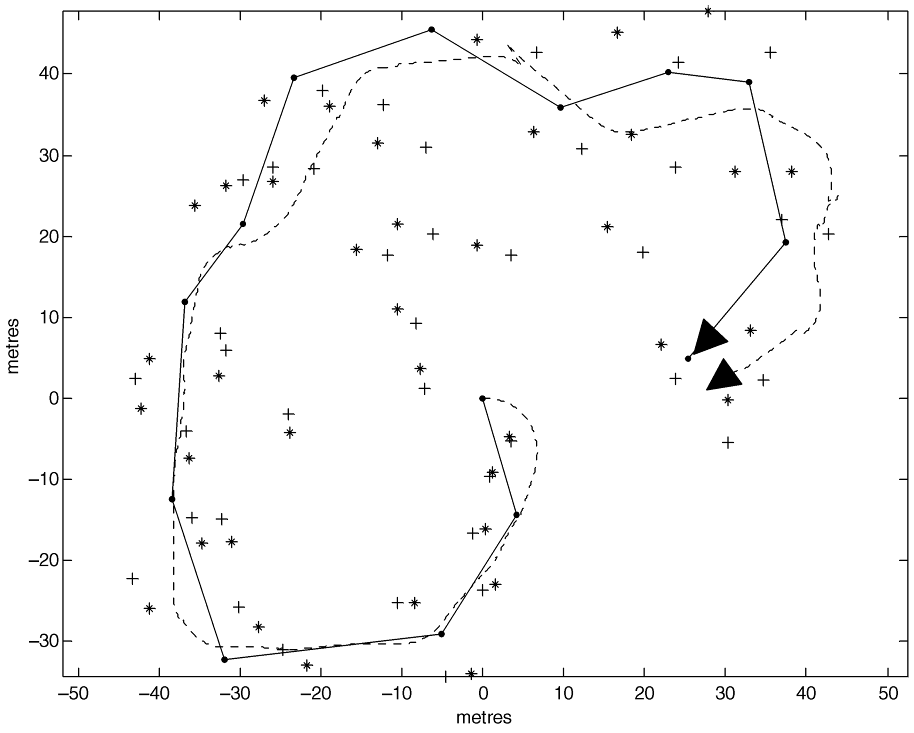
Now, if the new vector $\boldsymbol{u}_{i,G+1}$ can yield a smaller value for the fitness function, then this vector becomes the new $\boldsymbol{x}_{i,G+1}$ for the (G+1)th generation. Otherwise we keep $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$.

4. Performance Evaluation

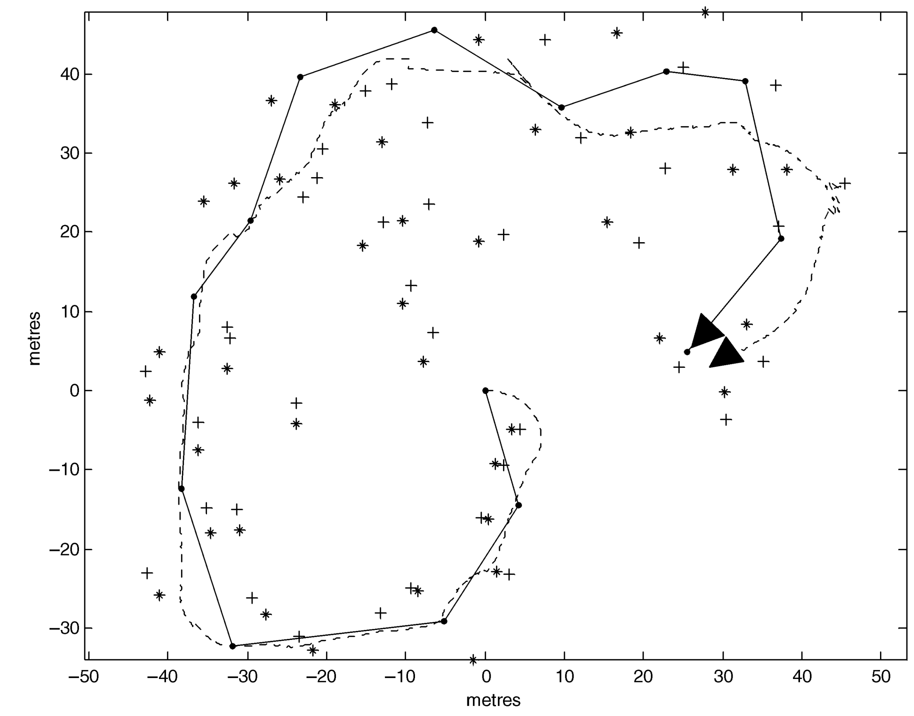
The proposed solution for the SLAM problems has been tested by creating an environment in simulation, utilizing the package available in ref. [13]. The environment created contains several waypoints through which the robot is directed to navigate and along its way it comes across several landmarks, whose positions should be correctly identified for building an accurate map. The package developed in ref. [13] is given for conventional EKF-based SLAM solutions, employing static **Q** and **R** matrices. It shows that the conventional EKF-based system gives satisfactory solution for the sensor statistics, specified as $\sigma_r = 0.1$ m and $\sigma_b = 1$ deg. and this result is demonstrated in Fig. 2. In the figure, the firm lines depict the ideal path that the robot is asked to traverse through the waypoints, while the dotted lines depict the SLAM estimated path traversed, based on estimated states of robot poses in each sampling instant. The stars (∗) depict the actual stationary landmark positions. The crosses (+) depict the positions of these landmarks estimated at the end of the test run. It can be seen that the performance of the conventional SLAM algorithm is quite satisfactory in this case.

However, the performance of the same system degrades considerably when we consider that these sensor statistics are wrongly known. Let us consider two sets of incorrect knowledge of sensor statistics as (a) $\sigma_r = 0.01$ m and $\sigma_b = 10.0$ deg. and (b) $\sigma_r = 0.01$ m and $\sigma_b = 15.0$ deg. Then the performances exhibited by the conventional EKF-based SLAM[13] are shown in Fig. 3(a) and (b). It can be seen that the estimated robot path deviates a lot from the ideal path and also the estimated positions of many landmarks are quite far away from their actual positions. However, when the proposed system was employed for each of these two case studies, the fuzzy supervision could improve the performance quite markedly, in each case, as depicted in Fig. 4(a) and (b). For the fuzzy supervised algorithm, the estimated robot paths deviated much less from the ideal robot paths. In this scheme, the free parameters of the fuzzy supervisor are learnt by implementing DE with $D = 11$ and employing binomial crossover. The variety of the DE algorithm employed is a popular variant, known as the "DE/rand/1" scheme.[11,12] However, this variant differs slightly from the original "DE/rand/1" scheme, because here the random selection of vectors is performed by shuffling the array containing the population so that a given vector does not get chosen twice in the same term contained in the perturbation expression.[14] It can also be seen that, for each case study, the estimated positions of the landmarks are in closer agreement with their actual positions, than the systems utilizing conventional EKF-based SLAM algorithms.

The results shown in Fig. 4 are obtained in the implementation phase, using the fuzzy supervisors trained by the DE algorithm, with the chosen control parameters $NP = 20$, $F = 0.1$, and $CR = 0.5$. Like most other stochastic global optimization methods, the performance of the DE strategy too varies with the choice of these free parameters. Hence proper choice or fitting of these parameters is crucial. According to the general guidelines proposed in ref. [11], for many applications, choices of $NP = 10 \times D$, $F \in [0.5, 1]$, and $CR \in [0, 1]$ but much lower than 1, are considered to be good choices. Among these factors, F is considered to



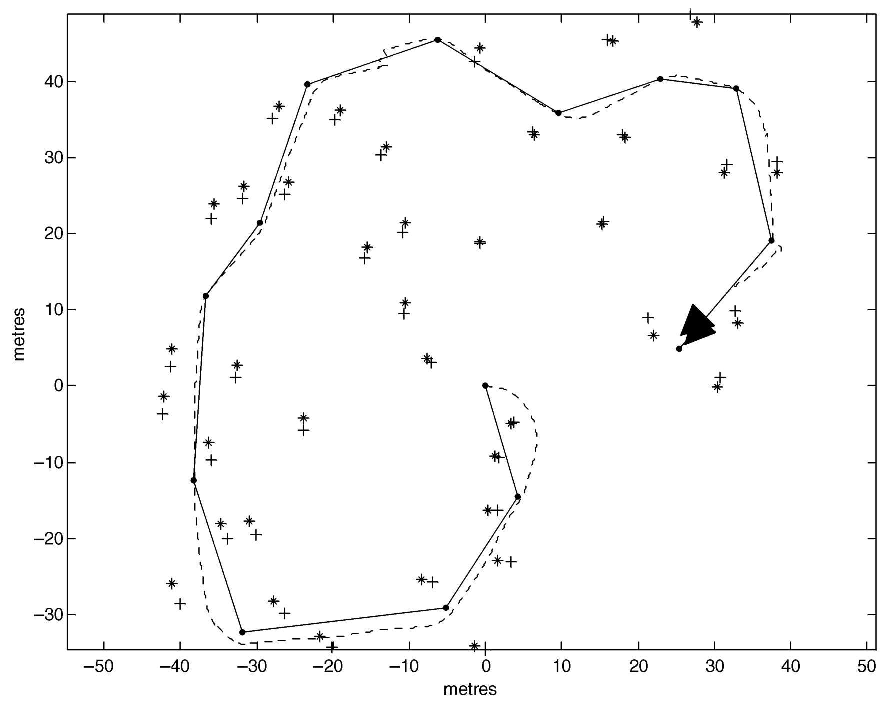
(a)



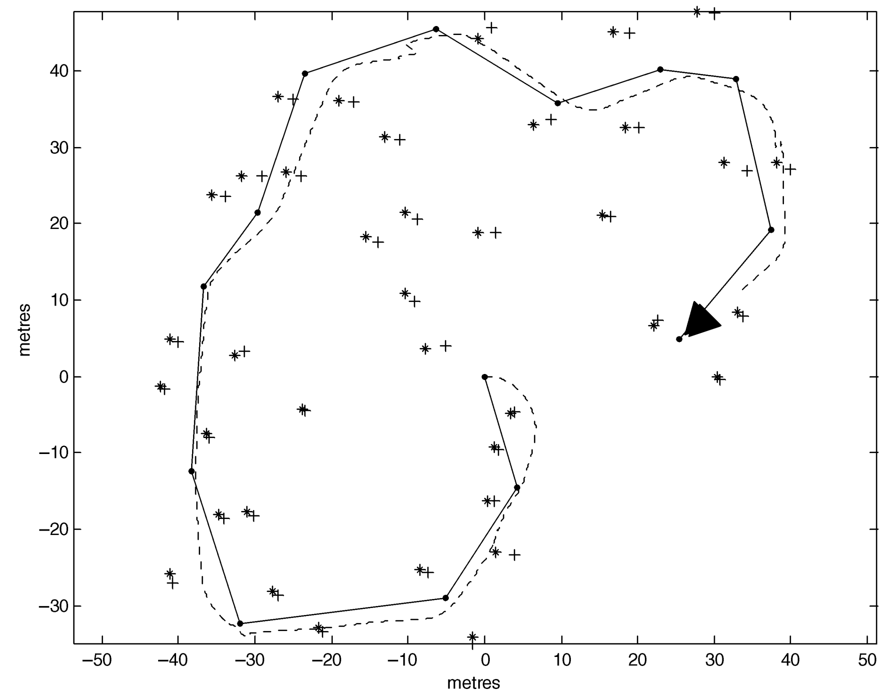
(b)

Fig. 3. Performance of the conventional EKF-based SLAM under incorrect knowledge of sensor statistics: (a) with ($\sigma_r = 0.01$ m and $\sigma_b = 10.0$ deg.) and (b) with ($\sigma_r = 0.01$ m and $\sigma_b = 15.0$ deg.).

be the most crucial control parameter and *NP* and *CR* are considered less crucial ones. Hence, in order to find the best performance of DE, it was considered to carry out simulations for various values of these control parameters and to observe their corresponding performances, for the case study with sensor statistics ($\sigma_r = 0.01$ m and $\sigma_b = 15.0$ deg.). At first, *NP* and *CR* are kept fixed at 20 and 0.5, respectively, and *F* is varied for a number of values in the range 0–1 and for each case the fuzzy supervisor was trained separately. Although, according to the general guideline, *NP* should have been chosen as $10 \times 11 = 110$, this would have increased the computational burden of the training procedure enormously. Hence, with the objective of keeping the computational burden reasonably low, the optimization procedure was



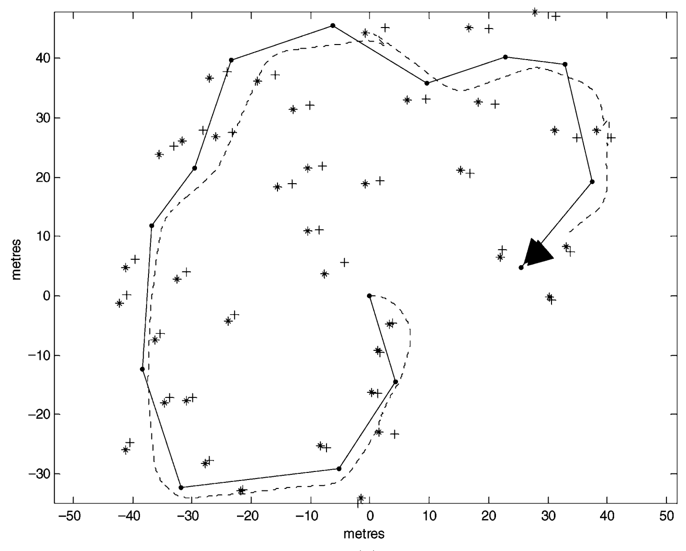
(a)



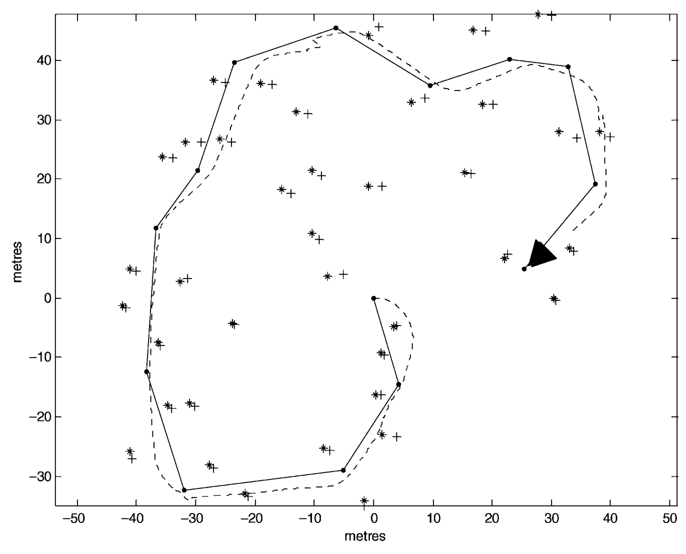
(b)

Fig. 4. Performance of the fuzzy supervised EKF-based SLAM, in implementation phase, under incorrect knowledge of sensor statistics: (a) with ($\sigma_r = 0.01$ m and $\sigma_b = 10.0$ deg.) and (b) with ($\sigma_r = 0.01$ m and $\sigma_b = 15.0$ deg.).

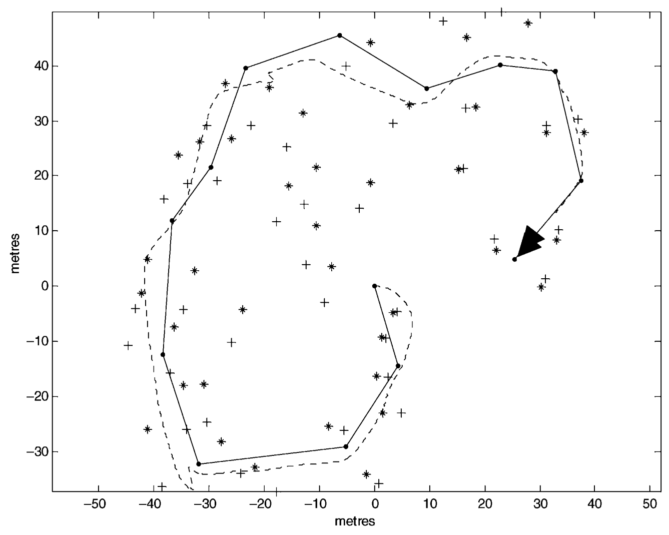
attempted with an *NP* value of 20. Here when F was varied, it was found that better and better performance of the overall system could be achieved in the implementation phase if we use smaller values of F. It was found that the best performance was achieved with $F = 0.1$ and with lower values of F the performance degraded a little while with higher values of F the degradation was significant. Figure 5(a)–5(c) shows the corresponding performance of the system in the implementation phase with the trained fuzzy supervision for $F = 0.05$, $F = 0.1$, and $F = 0.5$. Figure 6 shows the RMS (root mean square) errors in estimating $\hat{\mathbf{x}}$, in the implementation phase, at each sampling instant with an incremental movement of the robot, for this series of case studies with five representative values of F. It can be easily concluded that the training process conducted with $F = 0.1$ produced the best result for these experimentations.



(a)



(b)



(c)

Fig. 5. The implementation performance of the fuzzy supervised EKF-based SLAM, when the DE-based training was carried out with $NP = 20$, $CR = 0.5$, and (a) $F = 0.05$, (b) $F = 0.1$, and (c) $F = 0.5$.

With this value of F, one can proceed to determine the most suitable values of NP and CR. Keeping $F = 0.1$ and $CR = 0.5$, we varied NP for a series of values. The objective was to obtain a reasonable performance with as small a value of NP as practicable, so that the computational burden is kept minimum. Figure 7 shows the RMS errors in estimating $\hat{\mathbf{x}}$, in the implementation phase, at each sampling instant with an incremental movement of the robot, for this series of case studies with three representative values of $NP = 15$, 20, and 25. It was found that the best performance is obtained with $NP = 20$ and the performance degrades if we either increase or decrease the value of NP. Hence a value of $NP = 20$ was chosen for the training procedure. Next keeping $F = 0.1$ and $NP = 20$, CR was varied for a series of values. It was found that the variation of CR was not that critical in varying the training performance of the scheme. Figure 8 shows the similar plotting of RMS errors in estimating $\hat{\mathbf{x}}$, for this series of case studies with three representative values of $CR = 0.4$, 0.5, and 0.6. It was found that the best performance was obtained with $CR = 0.5$; although performances for other values of CR were quite similar in nature. Hence it could be concluded that the best set of control parameters of the DE for the training procedure of the fuzzy supervisor is obtained as $NP = 20$, $F = 0.1$, and $CR = 0.5$. Hence, using these parameters the fuzzy supervisor was trained for each case study of sensor statistics i.e. with (a) $\sigma_r = 0.01$ m and $\sigma_b = 10.0$ deg. and (b) $\sigma_r = 0.01$ m and $\sigma_b = 15.0$ deg. Figure 4(a) and (b) shows the performances of those case studies, in the implementation phase.

The performance of the DE-tuned fuzzy supervisor for EKF-based SLAM problems is then compared with another adaptive EKF-based SLAM algorithm, where the fuzzy supervisor for the adaptive EKF is trained by utilizing another similar contemporary popular stochastic global optimization algorithm, called PSO. The performance comparison is demonstrated for the sample case study with sensor statistics ($\sigma_r = 0.01$ m and $\sigma_b = 15.0$ deg.). PSO has gained much popularity in recent times, where several potential vectors,

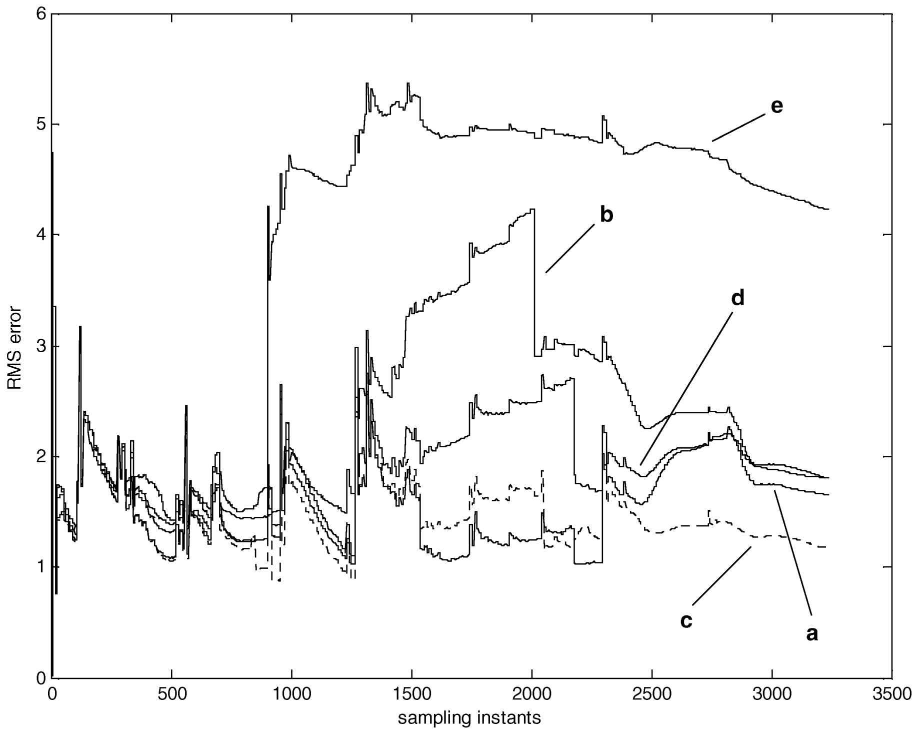


Fig. 6. The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the DE-based training was carried out with $NP = 20$, $CR = 0.5$, and (a) $F = 0.05$, (b) $F = 0.08$, (c) $F = 0.1$, (d) $F = 0.15$, and (e) $F = 0.5$.

called particles, are evaluated in a multidimensional space, for solving a global optimization problem. The algorithm is described in great detail in [15, 16] and in many other literatures. In this work, a very popular, standard variant of PSO is employed using linearly decreasing inertia weight. To make as uniform comparison between the DE-based and the PSO-based tuning algorithms for our problem as practicable, the following factors are taken into consideration: (i) identical number of candidate solutions or particles for each algorithm (i.e. 20), (ii) identical value of maximum number of iterations or generations for which the optimization algorithm is run each time (taken as 10 in this work), and (iii) identical

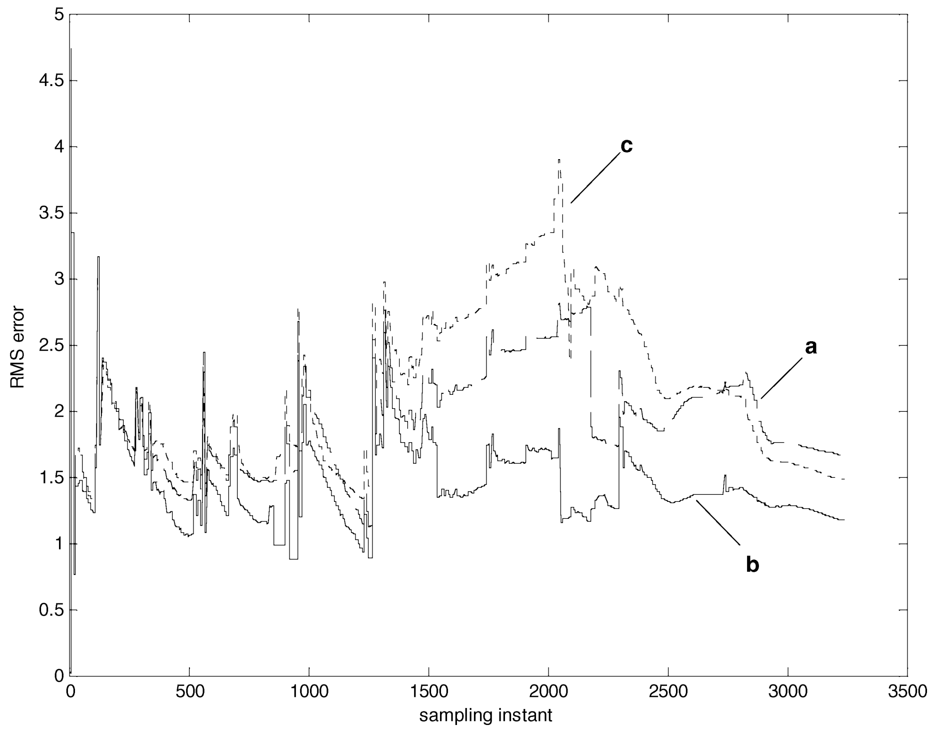


Fig. 7. The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the DE-based training was carried out with $F = 0.1$, $CR = 0.5$, and (a) $NP = 15$, (b) $NP = 20$, and (c) $NP = 25$.

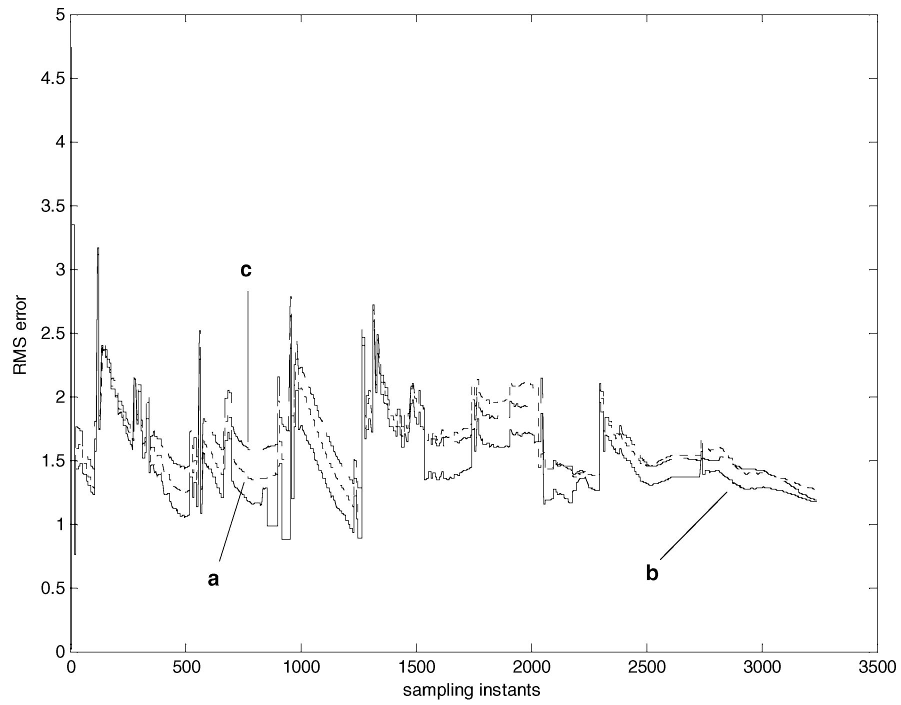


Fig. 8. The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the DE-based training was carried out with $F = 0.1$, $NP = 20$, and (a) $CR = 0.4$, (b) $CR = 0.5$, and (c) $CR = 0.6$.

range of initialization of each corresponding dimension of the initial population for each optimization algorithm. The PSO with inertia weight variation is normally known to perform well for benchmark optimization functions with initial inertia weight, W_{initial}, of 0.9 and slope of inertial weight of 2.5e-4. For our case study, we implemented PSO with $W_{\text{initial}} = 0.9$ and employed a series of both slow decrease and aggressive decrease in inertia weight. Figure 9 shows the corresponding performance of the PSO algorithm in terms of the RMS errors in estimating $\hat{\mathbf{x}}$, in the implementation phase, at each sampling instant with an incremental movement of the robot, for this series of case studies when the PSO-based training

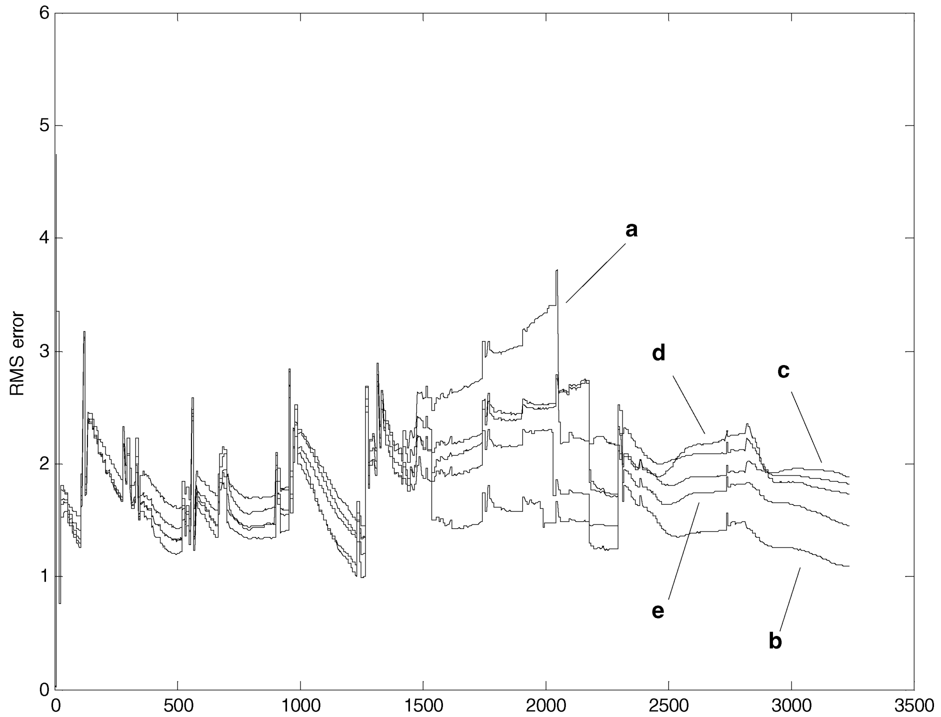


Fig. 9. The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the PSO-based training was carried out with the slope of inertia weight chosen as (a) 2.0e-4, (b) 2.5e-4, (c) 5.0e-4, (d) 4e-2, and (e) 5e-2.

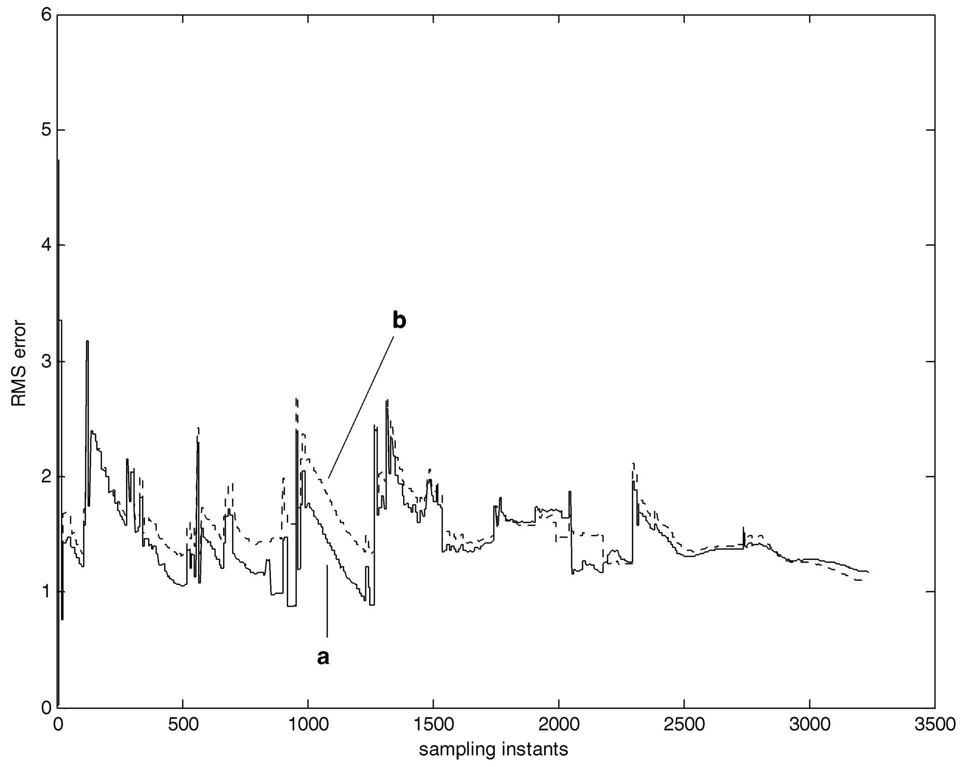


Fig. 10. Comparison of the estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the fuzzy supervisor is trained by (a) DE algorithm and (b) PSO algorithm.

procedure was conducted with slope of inertia weight having values 2.0e-4, 2.5e-4, 5.0e-4, 4e-2, and 5e-2. It was found that the best performance was indeed obtained with the universally known superior value of 2.5e-4. Figure 10 shows a similar comparison of estimation performance for the best PSO-tuned and best DE-tuned fuzzy supervisors for the adaptive EKF-based SLAM algorithm, for the case study under consideration. It can be seen that the performance of the DE-tuned algorithm gave less RMS errors in estimation, at most of the sampling instants. This procedure helps us demonstrating the usefulness of employing a DE-tuned fuzzy supervision for EKF-based SLAM problems.

5. Conclusion

The present paper proposed the development of a DE-optimized fuzzy supervisor for adaptive EKF-based SLAM algorithms, utilized for mobile robots. This system has been demonstrated to improve the performance under those test situations where the *a priori* knowledge about the sensor statistics is incorrect. The system can improve the performance by adapting the **R** matrix online. For useful performance of the fuzzy supervisor, the free parameters of the fuzzy-logic-based system are successfully learned by employing DE strategy offline. Two experiments in simulation have been carried out successfully which could confirm the usefulness of the proposed scheme.

The next step will be to explore the utility for successful implementation of the scheme for real-life slam problems. Such a problem will be a real challenging one, because of the presence of different types of features or landmarks, e.g. points, lines, and edges, and presence of landmarks with different shape irregularities in the practical situations. One possible solution can be to extract point features from such irregular environments by employing vision-based sensing where image registration can be successfully performed to extract good features, which can be successfully tracked in subsequent images acquired, i.e. in observation steps. The author wishes to take up this problem in near future.

Acknowledgment

This work was supported by All India Council for Technical Education under RPS scheme (Grant No. 8023/BOR/RPS-89/2006-07).

References

1. M. W. M. G. Dissanayake, P. Newman, S. Clark and H. F. Durrant-Whyte, “A solution to the simultaneous localization and map building (SLAM) problem,” *IEEE Tran. Rob. Automat.* **17**(3), 229–241 (Jun. 2001).
2. M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges,” *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico (2003).
3. R. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *Int. J. Rob. Res.* **5**(4), 56–68 (1986).
4. T. Bailey, Mobile Robot Localization and Mapping in Extensive Outdoor Environments *PhD Thesis* (University of Sydney, 2002).

5. A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Tran. Pattern Anal. Mach. Intell.* **24**(7), 865–880 (Jul. 2002).
6. J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map-building algorithm and real-time implementation," *IEEE Tran. Rob. Automat.* **17**(3), 242–257 (Jun. 2001).
7. R. K. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Tran. Automat. Control* **AC-15**(2), 175–184 (1970).
8. R. J. Fitzgerald, "Divergence of the Kalman filter," *IEEE Tran. Automat. Control* **AC-16**(6), 736–747 (1971).
9. K. Kobayashi, K. C. Cheok, K. Watanabe and F. Munekata, "Accurate differential global positioning system via fuzzy logic Kalman filter sensor fusion technique," *IEEE Tran. Indus. Electron.* **45**(3), 510–518 (Jun. 1998).
10. D. Loebis, R. Sutton, J. Chudley and W. Naeem, "Adaptive tuning of a Kalman filter via fuzzy logic for an intelligent AUV navigation system," *Control Eng. Pract.* **12**, 1531–1539 (2004).
11. R. Storn, "On the usage of differential evolution for function optimization" *Biennial Conference of the North American Fuzzy Information Processing Society* (NAFIPS), Berkley, USA, 19–22 June 1996, pp. 519–523.
12. R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution" *IEEE Conference on Evolutionary Computation*, Nagoya, Japan, 1996, pp. 842–844.
13. Available online. http://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm. Last accessed 24-June-2008.
14. Available online. http://www.icsi.berkeley.edu/~storn/code.html. Last accessed 24-Jun-2008.
15. M. Clerc and J. Kennedy, "The particle swarm-explosion, stability and convergence in a multidimensional complex space," *IEEE Tran. Evol. Comput.* **6**(1), 58–73 (Feb. 2002).
16. Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," *Proceedings of the IEEE Congress on Evolutionary Computation* (1999), pp. 1945–1950.