# Chapter 7
# Simultaneous Localization and Mapping (SLAM) in Mobile Robots*

**Abstract.** This chapter first introduces the concept of SLAM for navigation of mobile robots and then describes the extended Kalman filter (EKF) based SLAM algorithms in detail. Next we consider a more complex scenario where this EKF based SLAM algorithm is implemented in presence of incorrect knowledge of sensor statistics and discuss how fuzzy or neuro-fuzzy supervision can help in improving the estimation performance in such situations. In this context, we also discuss how evolutionary optimization strategies can be employed to automatically learn the free parameters of such neuro-fuzzy supervisors.

## 7.1 Introduction

The simultaneous localization and mapping (SLAM) problem has attracted significant attention from the research communities of the autonomous vehicles and mobile robots in the past two decades. The SLAM problem, essentially, consists of estimating the unknown motion of a moving platform iteratively, in an unknown environment and, hence, determining the map of the environment consisting of features (also known as landmarks) and the absolute location of the moving platform on the basis of each other's information [1]. This is known as a very complex problem as 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]. This situation arises when a vehicle does not have access to a global positioning system (GPS). Hence the complexity of the SLAM problem is manifold and requires a solution in a high dimensional space due to the mutual dependence of vehicle pose and the map estimates [3].

* This chapter is based on:

1) "A neuro-fuzzy assisted extended Kalman filter-based approach for Simultaneous Localization and Mapping (SLAM) problems," by Amitava Chatterjee and Fumitoshi Matsuno, which appeared in *IEEE Transactions on Fuzzy Systems*, vol. 15, issue 5, pp. 984-997, October 2007. © 2007 IEEE and
2) Amitava Chatterjee, "Differential evolution tuned fuzzy supervisor adapted extended kalman filtering for SLAM problems in mobile robots," *Robotica*, vol. 27, issue 3, pp. 411-423, May 2009, reproduced with permission from Cambridge University Press.

One of the oldest and popular approaches to solve the SLAM problem employs Kalman filter based techniques. Until now extensive research works have been reported employing EKF to address several aspects of the SLAM problem [1], [4-12]. Several successful applications of SLAM algorithms have been developed for indoor applications [13, 14], outdoor applications [7], underwater applications [15], underground applications [16] etc. An EKF based approach estimates and stores the robot pose and the feature positions within the map of the environment in the form of a complete state-vector and the uncertainties in these estimates are stored in the form of error covariance matrices. These covariance matrices also include cross-correlation terms signifying cross-correlation among feature/ landmark estimates. However, one of the well-known problems with the classical full EKF-based SLAM approach is that the computational burden becomes significantly high in the presence of a large number of features, because both the total state vector and the total covariance matrix become large in size. The later variations of researches on EKF based SLAMs have identified this problem as a key area and several improvements have so far been proposed [7, 9, 17-19]. Another key problem associated with EKF-based SLAM is the data association problem, which arises because several landmarks in the map may look similar. In those situations, different data association hypotheses can give rise to multiple, distinct looking maps and Gaussian distribution cannot be employed to represent such multi-modal distributions. This problem is usually solved by restricting the algorithm to associate the most likely data association, given the current robot map, on the basis of single measurement [1] or on the basis of multiple measurements [20]. The method of utilizing multiple measurements is a more robust method. Although several other data association algorithms have so far been developed, e.g. those in [21, 22], these algorithms have less significance as they cannot be implemented in real-time.

Some alternative approaches to solve SLAM problems have also been proposed which intend to implement some numerical algorithms, rather than employing the rigorous statistical methods as in EKF. Some of these schemes are based on the Bayesian approaches which can dispense with the important assumption in EKF (i.e. the uncertainties should be modeled by Gaussian distributions). Several such algorithms have been developed employing Sequential Monte Carlo (SMC) methods that employ the essence of particle filtering [2], [3], [23], [24]. Particle filtering technique can do away with a basic restriction of EKF algorithm that introduces an additional uncertainty by performing linearization of nonlinear models. However, in particle filtering based methods, it is expected that one should employ large number of particles so that it can contain a particle that can very closely resemble the true pose of the vehicle/robot at each sampling time instant [25]. How to develop an efficient SLAM algorithm, employing particle filtering with small enough number of particles, constitutes an important area of modern-day research. A significant leap in this direction is taken by the FastSLAM1.0 and FASTSLAM2.0 algorithms, which have successfully solved the issue of dimensionality for particle filter based SLAM problems [26]. Several other SLAM algorithms have also been successfully developed employing scan-matching technique where the map can efficiently be built by a graph of spatial relations amongst reference frames [7], [27].

It has been shown previously that the performance of an EKF process depends largely on the accuracy of the knowledge of process covariance matrix (**Q**) and measurement noise covariance matrix (**R**). An incorrect *a priori* knowledge of **Q** and **R** may lead to performance degradation [28] and it can even lead to practical divergence [29]. Hence adaptive estimation of these matrices becomes very important for online deployment. In [28], Mehra has reported a pioneering work on adaptive estimation of noise covariance matrices **Q** and **R** for Kalman filtering algorithm, based on correlation-innovations method, that can provide asymptotically normal, unbiased and consistent estimates of **Q** and **R** [35]. This algorithm is based on the assumption that noise statistics is stationary and the model under consideration is a time invariant one. Later several research works have been reported in the same direction, employing classical approaches, which have attempted adaptive estimation of **Q** and **R** [30-35]. In [30], a combination of an iterative algorithm and a stochastic approximation algorithm has been proposed to estimate **Q** and **R**. In [32] and [33], the problem domain has been expanded to allow time-variance in estimation of **Q** and **R**. A wonderful practical application of [28] has been reported in [34].

In the last ten years or so, there have also been several adaptive Kalman filtering algorithms proposed which employ fuzzy or neuro-fuzzy based techniques [36]-[39]. In [38], an input-output mapping problem, where output is corrupted by measurement noise, is solved by employing a neuro-fuzzy network to determine AR parameters of each operating point dependant ARMA model and then employing Kalman filter for the equivalent state-space representation of the system. In [36], fuzzy logic has been employed for simultaneous adaptive estimations of **Q** and **R** and in [37], fuzzy logic is employed to adapt the **R** matrix only, for a Kalman filter algorithm. In real world situations, it is quite perceptible that these information matrices, in the form of **Q** and **R**, may not be accurately known. Then the performance of the SLAM problem may get affected significantly.

The present chapter will first introduce the EKF-based stochastic SLAM algorithm in detail. Then the chapter will explore those situations for SLAM problems where the noise statistics information for the sensor is not known accurately. In those situations, we shall describe how neuro-fuzzy assisted EKF based SLAM algorithms can be effectively utilized [44, 45]. This will detail how a neuro-fuzzy model can be employed to assist the EKF-based SLAM algorithm to estimate **R** adaptively in each iteration. The chapter will also discuss how the free parameters of the neuro-fuzzy model can be learned using popular evolutionary optimization algorithms, for example, particle swarm optimization (PSO) [40] and differential evolution. The fuzzy adapted Kalman filter algorithms discussed in this chapter essentially implement a much complicated and sophisticated system compared to its predecessors mainly in two aspects:

i) For the SLAM problem, the situation is essentially very complex as the sizes of the state vector and hence the covariance matrix are time varying in nature. This is because, during the process of navigation, new landmarks are initialized in the state vector at different time instants (and, under some specific conditions, some existing landmarks may even be deleted) and hence these vector and matrix sizes will keep changing. The sizes of these matrices usually grow.

ii) The approaches discussed in this chapter uses a generalized method of learning the neuro-fuzzy model automatically. This is in stark contrast with previously developed systems which use carefully, manually chosen parameters for the fuzzy system(s) under consideration.

The chapter concludes with a detail, in-depth analysis of these SLAM algorithms where the results are presented for a variety of environmental situations i.e. with varying number of feature/landmark points and with several incorrectly known measurement noise statistics values.

## 7.2 Extended Kalman Filter (EKF) Based Stochastic SLAM Algorithm

*A. Hypotheses*

- The features under consideration are assumed to be 2-D point features
- The features are assumed to remain static i.e. they do not change their positions with time, in the map built
- There are uncertainties in control inputs, the steering angle command (*s*) and the velocity at which the rear wheel is driven (*w*), and these uncertainties are modeled using Gaussian distributions
- It is assumed that there is no uncertainty in the starting pose of the robot
- The incremental movement of the robot, between two successive sampling instants, is assumed to be linear in nature
- There are uncertainties in the range (*r*) and bearing (*θ*) measurements, and these uncertainties are modeled using Gaussian distributions
- The features are only characterized by their 2-D positions and no other characteristics, e.g. shape etc., is considered in this work

*B. The Algorithm*

An overview of the feature-map based SLAM employing EKF algorithm is presented now. An excellent description of the algorithm can also be obtained in [6], [7]. An EKF is employed for state estimation in those situations where the process is governed by nonlinear dynamics and/or involves nonlinear measurement relationships. The method employs linearization about the filter's estimated trajectory, which is continuously updated in accordance with the state estimates obtained from the measurements [43]. The state transition can be modeled by a nonlinear function **f**(•) and the observation or measurement of the state can be modeled by a nonlinear function **h**(•), given as:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{q}_k \tag{7.1}$$

and

$$\mathbf{z}_{k+1} = \mathbf{h}(\mathbf{x}_{k+1}) + \mathbf{r}_{k+1} \tag{7.2}$$

where $\mathbf{x}_k$ is the $(n \times 1)$ process state vector at sampling instant $k$, $\mathbf{z}_k$ is the $(m \times 1)$ measurement vector at sampling instant $k$ and $\mathbf{u}_k$ is the control input. The random variables $\mathbf{q}_k$ and $\mathbf{r}_k$ represent Gaussian white process noise and measurement noise respectively and $\mathbf{P}_k$, $\mathbf{Q}_k$ and $\mathbf{R}_k$ represent the covariance matrices for $\mathbf{x}_k$, $\mathbf{q}_k$ and $\mathbf{r}_k$ respectively.

In case of the SLAM problem, the state vector $\mathbf{x}$ is composed of the vehicle states $\mathbf{x}_v$ and the landmarks' states $\mathbf{x}_m$. Hence the estimates of the total state vector $\mathbf{x}$, maintained in the form of its mean vector $\hat{\mathbf{x}}$ and the corresponding total error covariance matrix $\mathbf{P}$, is given as:

$$\hat{\mathbf{x}} = [\hat{\mathbf{x}}_v^T \ \ \hat{\mathbf{x}}_m^T]^T \tag{7.3}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_v & \mathbf{P}_{vm} \\ \mathbf{P}_{vm}^T & \mathbf{P}_m \end{bmatrix} \tag{7.4}$$

where $\hat{\mathbf{x}}_v$ = the mean estimate of the robot/vehicle states (represented by its pose),

$\mathbf{P}_v$ = error covariance matrix associated with $\hat{\mathbf{x}}_v$ ,

$\hat{\mathbf{x}}_m$ = mean estimate of the feature positions and

$\mathbf{P}_m$ = error covariance matrix associated with $\hat{\mathbf{x}}_m$ .

The robot/vehicle pose is defined with respect to an arbitrary base Cartesian coordinate frame. The features or landmarks are considered to be 2-D point features. It is assumed that there are $n$ such static, point features observed in the map. Then,

$$\hat{\mathbf{x}}_v = [\hat{x}_v \ \hat{y}_v \ \ \hat{\varphi}_v]^T \tag{7.5),}$$

$$\mathbf{P}_v = \begin{bmatrix} \sigma^2_{x_v x_v} & \sigma^2_{x_v y_v} & \sigma^2_{x_v \varphi_v} \\ \sigma^2_{x_v y_v} & \sigma^2_{y_v y_v} & \sigma^2_{y_v \varphi_v} \\ \sigma^2_{x_v \varphi_v} & \sigma^2_{y_v \varphi_v} & \sigma^2_{\varphi_v \varphi_v} \end{bmatrix} \tag{7.6),}$$

$$\hat{\mathbf{x}}_m = [\hat{x}_1 \ \hat{y}_1 \ \ldots\ldots \ \hat{x}_n \ \hat{y}_n]^T \tag{7.7}$$

and

$$\mathbf{P}_m = \begin{bmatrix} \sigma^2_{x_1 x_1} & \sigma^2_{x_1 y_1} & \cdots & \sigma^2_{x_1 x_n} & \sigma^2_{x_1 y_n} \\ \sigma^2_{x_1 y_1} & \sigma^2_{y_1 y_1} & \cdots & \sigma^2_{y_1 x_n} & \sigma^2_{y_1 y_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma^2_{x_1 x_n} & \sigma^2_{y_1 x_n} & \cdots & \sigma^2_{x_n x_n} & \sigma^2_{x_n y_n} \\ \sigma^2_{x_1 y_n} & \sigma^2_{y_1 y_n} & \cdots & \sigma^2_{x_n y_n} & \sigma^2_{y_n y_n} \end{bmatrix} \tag{7.8}$$

The map is defined in terms of the position estimates of these static features and $\mathbf{P}_{vm}$ in (7.4) maintains the robot-map correlation. The off-diagonal elements of $\mathbf{P}_m$

signify the cross-correlation and hence interdependence of information among the features themselves. The system is initialized assuming that there is no observed feature as yet, the base Cartesian coordinate frame is aligned with the robot's starting pose and there is no uncertainty in the starting pose of the robot. Mathematically speaking, $\hat{\mathbf{x}} = \hat{\mathbf{x}}_v = \mathbf{0}$ and $\mathbf{P} = \mathbf{P}_v = \mathbf{0}$.

As the robot starts moving, $\hat{\mathbf{x}}_v$ and $\mathbf{P}_v$ become non-zero values. In subsequent iterations, when the first observation is carried out, new features are expected to be initialized and $\hat{\mathbf{x}}_m$ and $\mathbf{P}_m$ appear for the first time. This increases the size of $\hat{\mathbf{x}}$ and $\mathbf{P}$ and the entries of $\hat{\mathbf{x}}$ vector and $\mathbf{P}$ matrix are re-calculated. This process is continued iteratively.

*i) Time Update ("Predict") Step*

Here, it is assumed that the control input vector $\mathbf{u}$, under the influence of which the robot moves, is constituted of two control inputs, the steering angle command ($s$) and the velocity at which the rear wheel is driven ($w$). Hence, $\mathbf{u} = [w \ s]^T$. So the state estimates can be obtained by employing wheel encoder odometry and the robot kinematic model. The control inputs $w$ and $s$ must be considered with their uncertainties involved (e.g. uncertainties due to wheel slippage, incorrect calibration of vehicle controller) and these are modeled as Gaussian variations in $w$ and $s$ from their nominal values. Hence, the prediction step calculates the projections of the state estimates and the error covariance estimates from sampling instant $k$ to ($k$+1), given as:

$$\hat{\mathbf{x}}_{k+1}^{-} = \mathbf{f}(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k) = \begin{bmatrix} \hat{\mathbf{x}}_{v_{k+1}}^{-} \\ \hat{\mathbf{x}}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_v(\hat{\mathbf{x}}_{v_k}, \hat{\mathbf{u}}_k) \\ \hat{\mathbf{x}}_m \end{bmatrix} \tag{7.9}$$

$$\mathbf{P}_{k+1}^{-} = \begin{bmatrix} \nabla\mathbf{f}_{\mathbf{x}_{v_k}} \mathbf{P}_{v_k} \nabla\mathbf{f}_{\mathbf{x}_{\mathbf{v}_k}}^T + \nabla\mathbf{f}_{\mathbf{u}_k} \mathbf{U}_k \nabla\mathbf{f}_{u_k}^T & \nabla\mathbf{f}_{\mathbf{x}_{v_k}} \mathbf{P}_{vm_k} \\ (\nabla\mathbf{f}_{\mathbf{x}_{v_k}} \mathbf{P}_{vm_k})^T & \mathbf{P}_m \end{bmatrix} \tag{7.10}$$

where $\mathbf{f}_v$ estimates the robot pose on the basis of the motion model and the control inputs. Based on the odometric equation of the mobile robot under consideration here, which assumes that the incremental movement of the robot is linear in nature, $\mathbf{f}_v$ can be represented as [42]:

$$\hat{\mathbf{x}}_{v_{k+1}}^{-} = \begin{bmatrix} \hat{x}_{v_{k+1}}^{-} \\ \hat{y}_{v_{k+1}}^{-} \\ \hat{\varphi}_{v_{k+1}}^{-} \end{bmatrix} = \mathbf{f}_v(\hat{\mathbf{x}}_{v_k}, \hat{\mathbf{u}}_k) = \begin{bmatrix} \hat{x}_{v_k} + w_k * \Delta t * \cos(s_k + \hat{\varphi}_{v_k}) \\ \hat{y}_{v_k} + w_k * \Delta t * \sin(s_k + \hat{\varphi}_{v_k}) \\ \hat{\varphi}_{v_k} + w_k * \Delta t * \dfrac{\sin(s_k)}{WB} \end{bmatrix} \tag{7.11}$$

where, *WB* represents the wheelbase of the robot and $\Delta t$ is the sampling time. The Jacobians and $\mathbf{U}_k$, the covariance matrix of $\mathbf{u}$ are given as:

$$\nabla \mathbf{f}_{\mathbf{x}_{v_k}} = \left.\frac{\partial \mathbf{f}_{v_k}}{\partial \mathbf{x}_{v_k}}\right|_{(\hat{\mathbf{x}}_{v_k}, \hat{\mathbf{u}}_k)} \quad (7.12), \quad \nabla \mathbf{f}_{\mathbf{u}_k} = \left.\frac{\partial \mathbf{f}_{v_k}}{\partial \mathbf{u}_k}\right|_{(\hat{\mathbf{x}}_{v_k}, \hat{\mathbf{u}}_k)} \quad (7.13), \quad \mathbf{U} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_s^2 \end{bmatrix} \tag{7.14}$$

Here, $\hat{\mathbf{x}}_m$ and $\mathbf{P}_m$ in (7.9) and (7.10) remain constant with time, as the features are assumed to remain stationary with time.

*ii) Measurement Update ("Correct") Step*

Let us assume that we observe a feature, which already exists in the feature map, whose position is denoted by that of the *i*th feature i.e. $(\hat{x}_i, \hat{y}_i)$. For the system under consideration [7], [42], it is assumed that the feature observation is carried out using 2-D scanning range laser (SICK PLS), a range-bearing sensor, which nowadays is very popular in mobile robot navigation, for distance measurement. It is assumed that the laser range scanner is mounted on the front bumper of the vehicle and the laser returns a 180° planar sweep of range measurements in 0.5° intervals. The range resolution of such a popular sensor is usually about ±50 mm. In this context, it should be mentioned that the vehicle is also assumed to be equipped with wheel and steering encoders. The distance measured, in polar form, gives the relative distance between each feature and the scanner (and hence the vehicle). Let this feature be measured in terms of its range ($r$) and bearing ($\theta$) relative to the observer, given as:

$$\mathbf{z} = [r \quad \theta]^T \tag{7.15}$$

The uncertainties in these observations are again modeled by Gaussian variations and let $\mathbf{R}$ be the corresponding observation/measurement noise covariance matrix given as:

$$\mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} \tag{7.16}$$

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

$$\hat{\mathbf{z}}_{i_k} = \mathbf{h}_i(\hat{\mathbf{x}}_k) = \begin{bmatrix} \sqrt{(\hat{x}_i - \hat{x}_{v_k})^2 + (\hat{y}_i - \hat{y}_{v_k})^2} \\ \arctan\left(\dfrac{\hat{y}_i - \hat{y}_{v_k}}{\hat{x}_i - \hat{x}_{v_k}}\right) - \hat{\varphi}_{v_k} \end{bmatrix} \tag{7.17}$$

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

$$\nu_{i_{k+1}} = \mathbf{z}_{k+1} - \mathbf{h}_i(\hat{\mathbf{x}}_{k+1}^{-}) \tag{7.18}$$

$$\mathbf{S}_{i_{k+1}} = \nabla \mathbf{h}_{\mathbf{x}_{k+1}} \mathbf{P}_{k+1}^{-} \nabla \mathbf{h}_{\mathbf{x}_{k+1}}^{T} + \mathbf{R}_k \tag{7.19}$$

$$\mathbf{W}_{i_{k+1}} = \mathbf{P}_{k+1}^{-} \nabla \mathbf{h}_{\mathbf{x}_{k+1}}^{T} \mathbf{S}_{i_{k+1}}^{-1} \tag{7.20}$$

where $\nu_i$ denotes the innovation of the observation for this *i*th landmark and $\mathbf{S}_i$ the associated innovation covariance matrix. The Jacobian $\nabla \mathbf{h}_{\mathbf{x}_{k+1}}$ is given as:

$$\nabla \mathbf{h}_{\mathbf{x}_{k+1}} = \left. \frac{\partial \mathbf{h}_i}{\partial \mathbf{x}_k} \right|_{\hat{\mathbf{x}}_{k+1}^{-}} \tag{7.21}$$

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

$$\hat{\mathbf{x}}_{k+1}^{+} = \hat{\mathbf{x}}_{k+1}^{-} + \mathbf{W}_{i_{k+1}} \nu_{i_{k+1}} \tag{7.22}$$

$$\mathbf{P}_{k+1}^{+} = \mathbf{P}_{k+1}^{-} - \mathbf{W}_{i_{k+1}} \mathbf{S}_{i_{k+1}} \mathbf{W}_{i_{k+1}}^{T} \tag{7.23}$$

Here it should be remembered that in addition to the process and measurement uncertainties, there is an additional uncertainty due to linearization involved in the formulation of an EKF. The "time update" and "measurement update" equations are obtained by employing linearization of nonlinear functions **f(•)** and **h(•)** about the point of the state mean. This linearization is obtained by employing a Taylor series like expansion and neglecting all terms which are of higher order than the first order term in the series. This manner of approximating a nonlinear system by a first order derivative introduces this additional source of uncertainty in EKF algorithm. In fact, for highly nonlinear functions, these linearized transformations cannot sufficiently accurately approximate correct covariance transformations and this may lead to highly inconsistent uncertainty estimate. Under those situations unscented transform may provide more accurate results.

*iii) Initialization of a new feature and deletion of an old feature*

During this iterative procedure of performing prediction and update steps recursively, it is very likely that observations of new features are made time to time. Then these new features should be initialized into the system by incorporating their 2-D position coordinates in the augmented state vector and accordingly modifying the covariance matrix. These features, identified by the

LRS, may correspond to points, lines, corners, edges etc. In this work, we have considered that the features are point like features, each representing a unique distinct point in the two-dimensional map of the environment. Resorting to the mathematical computations as shown in [7], these new $\hat{\mathbf{x}}_k^+$ and $\mathbf{P}_k^+$ can be calculated as:

$$\hat{\mathbf{x}}_k^+ = \begin{bmatrix} \hat{\mathbf{x}}_k \\ \mathbf{f}_f(\hat{\mathbf{x}}_{v_k}, \mathbf{z}_k) \end{bmatrix} \tag{7.24}$$

$$\mathbf{P}_k^+ = \begin{bmatrix} \mathbf{P}_{v_k} & \mathbf{P}_{vm_k} & \mathbf{P}_{v_k} \nabla \mathbf{f}_{\mathbf{f}_{v_k}}^T \\ \mathbf{P}_{vm_k}^T & \mathbf{P}_m & \mathbf{P}_{vm_k}^T \nabla \mathbf{f}_{\mathbf{f}_{v_k}}^T \\ \nabla \mathbf{f}_{\mathbf{f}_{v_k}} \mathbf{P}_{v_k} & \nabla \mathbf{f}_{\mathbf{f}_{v_k}} \mathbf{P}_{vm_k} & \nabla \mathbf{f}_{\mathbf{f}_{v_k}} \mathbf{P}_{v_k} \nabla \mathbf{f}_{\mathbf{f}_{v_k}}^T + \nabla \mathbf{f}_{\mathbf{f}_{z_k}} \mathbf{R}_k \nabla \mathbf{f}_{\mathbf{f}_{z_k}}^T \end{bmatrix} \tag{7.25}$$

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

$$\nabla \mathbf{f}_{\mathbf{f}_{v_k}} = \left.\frac{\partial \mathbf{f}_f}{\partial \mathbf{x}_{v_k}}\right|_{(\hat{\mathbf{x}}_{v_k}, \mathbf{z}_k)}, \quad \nabla \mathbf{f}_{\mathbf{f}_{z_k}} = \left.\frac{\partial \mathbf{f}_f}{\partial \mathbf{z}_k}\right|_{(\hat{\mathbf{x}}_{v_k}, \mathbf{z}_k)} \tag{7.26}$$

The deletion of unreliable features is a relatively simple matter. We only need to delete the relevant row entries from the state vector and the relevant row and column entries from the covariance matrix.

Now, it is quite common that when an observation step is carried out, there will be multiple number of landmarks visible at the same time and hence, several independent observations will be carried out. In our system, we have assumed that a batch of such observations is available at once (i.e. $\mathbf{z} = [r_1,\ \theta_1, \cdots r_n,\ \theta_n\ ]^T$) and updates are carried out in batches. This is in conformation with the arguments placed in [7] which indicate that an EKF algorithm tends to perform better update steps for SLAM algorithms, if the innovation vector $\nu$ consists of multiple observations simultaneously. Hence, in the context of this batch mode of observation and update procedure, the corresponding SLAM algorithm is based on composite $\nu$, $\mathbf{S}$ and $\mathbf{W}$ vectors/matrices and the sizes of these vectors/matrices keep changing with time because at any instant of observation, the total number of visible landmarks keep changing.

## 7.3 Neuro-fuzzy Assistance for EKF Based SLAM

Most of the works reported in the area of adaptive Kalman filters have so far concentrated on utilizing new statistical information from innovation sequence to correct the estimation of the states. Our approach for adapting the EKF is based on the innovation adaptive estimation (IAE) approach, which was originally proposed in [28] and later utilized in combination with fuzzy logic in [37]. 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 $k$th instant, and utilizing this new information to correct the estimations/predictions already made. The adaptation strategy is based on the objective of reducing mismatch between the theoretical covariance of the innovation sequences ($\mathbf{S}_k$) and the corresponding actual covariance of the innovation sequences ($\hat{\mathbf{C}}_{Innk}$). In our SLAM algorithm, $\mathbf{S}_k$ is calculated using (7.19) where the right hand side of the equation is made consistent with the concept of batch mode of observation and update. $\hat{\mathbf{C}}_{Innk}$ can be calculated as:

$$\hat{\mathbf{C}}_{Innk} = \nu_k \, \nu_k^T \tag{7.27}$$

where $\nu_k$ denotes the augmented innovation sequence, made consistent with the batch mode. According to [37], this covariance should be calculated on the basis of a moving average of $\nu_k \, \nu_k^T$ over an appropriate moving estimation window of size $M$. However, for the SLAM problem, the size of the augmented $\nu_k$ keeps changing from time to time. This is because it is dependent on the number of landmarks observed in any given *observation and update step*, which were all observed at least once before. Hence we employ (7.27) to calculate $\hat{\mathbf{C}}_{Innk}$ rather than using a moving average. Therefore, the mismatch at the $k$th instant, is given as:

$$\Delta\hat{\mathbf{C}}_{Innk} = \hat{\mathbf{C}}_{Innk} - \mathbf{S}_k \tag{7.28}$$

Our objective is to minimize this mismatch employing fuzzy logic. This is carried out, by employing a one-input-one-output neuro-fuzzy system for each diagonal element of the $\Delta\hat{\mathbf{C}}_{Innk}$ matrix. These fuzzy rules are employed to adapt the $\mathbf{R}$ matrix, so that the sensor statistics is adapted for subsequent reduction in mismatch $\Delta\hat{\mathbf{C}}_{Innk}$. The complete EKF-based SLAM algorithm, employing the neuro-fuzzy assistance, is presented in algo. 7.1. The system is designed with a sampling time of 25 msec. between successive control input signals.

```
1. IF All waypoints are traversed, THEN Stop ENDIF.
2. Compute distance of the robot from the current waypoint.
      IF (distance < minimum distance allowed from any waypoint),
         THEN switch to next waypoint as the current waypoint ENDIF.
3. Compute change in steering angle ($\Delta s$) to point towards the current waypoint and then, new value of steering angle ($s$) (satisfying the constraints of max. rate of steering change ($\Delta s_{max}$) and max. steering angle ($s_{max}$)).
4. Move the robot and determine its actual pose.
5. Perform EKF prediction step, in accordance with (7.7) to (7.10).
6. IF (Time_for_Observation is TRUE), THEN go to step 7. ELSE go to step 1. ENDIF.
7. Determine the set of visible landmarks from the current actual robot position. Compute actual range-bearing observation for each of them. Separate those observations based on already observed landmarks and newly observed landmarks (if any).
8. Predict range-bearing observations, for already observed landmarks in step 7, on the basis of augmented total state vector, predicted in step 5.
9. Compute augmented innovation sequence ($\nu$) for already observed landmarks, on the basis of actual and predicted observations, employing (7.14), adapted for batch-mode situations.
10. Compute corresponding augmented measurement noise covariance matrix $\mathbf{R}$ (utilizing the original [$2 \times 2$] $\mathbf{R}$ matrix) and augmented linearized observation model $\mathbf{h}$, adapted for batch-mode situations.
11. Compute augmented $\mathbf{S}$, on the basis of the augmented $\mathbf{R}$ and $\mathbf{h}$ and employing (7.15), adapted for batch-mode situations.
12. Update the a posterior state estimate vector and error covariance matrix, according to (7.18) and (7.19).
13. Compute $\hat{\mathbf{C}}_{Innk}$ and $\Delta\hat{\mathbf{C}}_{Innk}$, according to (7.23) and (7.24) respectively, and determine the size of $\Delta\hat{\mathbf{C}}_{Innk}$, i.e. $[\Delta\hat{\mathbf{C}}_{rows,} \Delta\hat{\mathbf{C}}_{cols}]$.
14. Determine the absolute maximum value of mismatch among the range observations ($\Delta\hat{\mathbf{C}}_{Innk\_range\_mismatch}$) and the bearing observations ($\Delta\hat{\mathbf{C}}_{Innk\_bearing\_mismatch}$) separately from the corresponding diagonal entries of the $\Delta\hat{\mathbf{C}}_{Innk}$ matrix.
15. FOR $j = 1$ to $\Delta\hat{\mathbf{C}}_{rows}$,
      Normalize the corresponding diagonal entry $\Delta\hat{\mathbf{C}}_{Innk}(j, j)$ by the appropriate
      $\Delta\hat{\mathbf{C}}_{Innk\_range\_mismatch}$ or $\Delta\hat{\mathbf{C}}_{Innk\_bearing\_mismatch}$.
```

```
        Determine the corresponding $\Delta\mathbf{R}(j,j)$ output from the NFS, with the
        normalized $\Delta\hat{\mathbf{C}}_{Innk}(j,j)$ input to it.
    ENDFOR
16. Determine $\Delta\sigma_r^2$ as a mean of those $\Delta\mathbf{R}(j,j)$ entries, which correspond to
    range measurements.
17. Determine $\Delta\sigma_\theta^2$ as a mean of those $\Delta\mathbf{R}(j,j)$ entries, which correspond to
    bearing measurements.
18. Adapt the original 2×2 R matrix as: $\mathbf{R}_k = \mathbf{R}_{k\text{-}1} + \begin{bmatrix} \Delta\sigma_r^2 & 0 \\ 0 & \Delta\sigma_\theta^2 \end{bmatrix}$.
19. IF (new feature(s) observed in step 7),
    THEN augment state vector and error covariance matrix, according to (7.20),
        (7.21) and (7.22).
    ENDIF
20. Go to step 1.
```

**Algo. 7.1.** The neuro-fuzzy assisted EKF based SLAM algorithm

From algo. 7.1, it can be seen that each Neuro-Fuzzy System (NFS) employs a nonlinear mapping of the form: $\Delta\mathbf{R}(j,j) = f_{NFS}(\Delta\hat{\mathbf{C}}_{Innk}(j,j))$ where $\Delta\mathbf{R}(j,j)$ corresponds to an adaptation recommended for the corresponding diagonal element of the augmented measurement noise covariance matrix **R** matrix, computed according to the batch-mode situation. This augmented matrix is calculated each time an iteration enters into the *observe and update step* and its size is determined on the basis of the total landmarks visible in the *observe step*. To make it consistent with the batch of observed landmarks that were already visited at least once earlier, the size of this augmented **R** 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:

$$\text{augmented } \mathbf{R} = \begin{bmatrix} \sigma_r^2 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & \sigma_\theta^2 & \cdots & \cdots & \cdots & 0 & 0 \\ \vdots & \vdots & \sigma_r^2 & 0 & \cdots & \vdots & \vdots \\ \vdots & \vdots & 0 & \sigma_\theta^2 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & \sigma_r^2 & 0 \\ 0 & 0 & \cdots & \cdots & \cdots & 0 & \sigma_\theta^2 \end{bmatrix} \tag{7.29}$$

Here, $\sigma_r^2$ and $\sigma_\theta^2$ correspond to the sensor statistics computed for that iteration. It can be seen that the augmented **R** matrix comprises of diagonal elements only and all the off-diagonal elements are considered to be zero. This is in conformation with our assumptions presented beforehand, in section 7.2, that the range and the bearing measurements are independent of each other and there is no cross-correlation between these measurements. The size of this augmented **R** matrix keeps changing in different iterations, as the number of already visited landmarks observed again in a given iteration keeps varying from iteration to iteration. The size of this augmented **R** is consistent with that of the $\hat{\mathbf{C}}_{Innk}$ and hence, $\Delta\hat{\mathbf{C}}_{Innk}$.

With the idea of implementing the same NFS for each and every diagonal element of the augmented **R** matrix, we employ normalized input for each NFS. The NFS practically employs three fuzzy ***IF-THEN*** rules of the form:

***IF*** $\Delta\hat{\mathbf{C}}_{Innk}(j, j)$ is N ***THEN*** $\Delta\mathbf{R}(j, j) = w_1$,

***IF*** $\Delta\hat{\mathbf{C}}_{Innk}(j, j)$ is Z ***THEN*** $\Delta\mathbf{R}(j, j) = w_2$ and

***IF*** $\Delta\hat{\mathbf{C}}_{Innk}(j, j)$ is P ***THEN*** $\Delta\mathbf{R}(j, j) = w_3$.

$w_1$, $w_2$ and $w_3$ indicate the amount of fuzzy adaptation recommended in form of a diagonal element of the $\Delta$**R** matrix, depending on the nature of the fuzzified mismatch in the corresponding diagonal element of the $\Delta\hat{\mathbf{C}}_{Innk}$ matrix. However, the order of mismatch may be different for range and bearing observations and this may depend on how poorly (or accurately) the sensor statistics for range and bearing observations are individually known. Hence we employ normalized inputs corresponding to range and bearing observations separately, on the basis of appropriate computations of $\Delta\hat{\mathbf{C}}_{Innk\_range\_mismatch}$ and $\Delta\hat{\mathbf{C}}_{Innk\_bearing\_mismatch}$, as given in algo. 7.1. Then with these normalized inputs, the NFS enables us to compute $\Delta\mathbf{R}(j, j)$ for each diagonal entry. Finally we compute the adaptations i.e. $\Delta\sigma_r^2$ and $\Delta\sigma_\theta^2$ required for the original [2 × 2] **R** matrix on the basis of 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 kept ready for the next appropriate iteration, when EKF will enter the *observation and update step*, and will be utilized for subsequent formation of augmented **R** matrix and so on. Then, each *observation and update step* is concluded by augmenting the state vector and the corresponding covariance matrix, by employing (7.24)-(7.26), if there are new feature(s) observed during this *observation step*.

## 7.4 The Neuro-fuzzy Architecture and Its Training Methodology Employing Particle Swarm Optimization (PSO)

### *7.4.1 Architecture of the Neuro-fuzzy Model*

The neuro-fuzzy model has been developed as a one-input-one-output system. The four-layer architecture is shown in Fig. 7.1. Let $u_i^l$ and $O_i^l$ respectively denote the input to and output from the *i*th node of the *l*th layer.

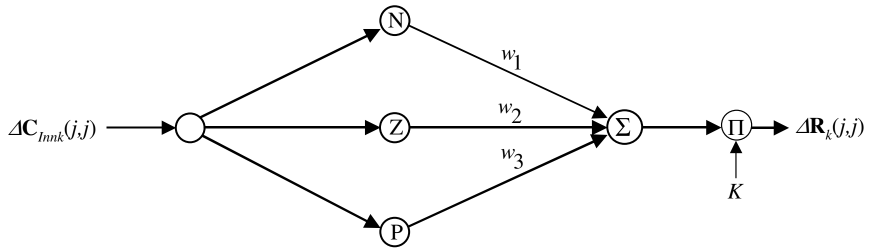


**Fig. 7.1.** Four-layer architecture of the proposed neuro-fuzzy system. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).

**1) Layer 1: Input Layer**

This layer comprises a single node, signifying the single input variable. The input-output relation of this node is:

$$O^1 = u^1 = \Delta\hat{\mathbf{C}}_{Innk}(j,j) \tag{7.30}$$

**2) Layer 2: Membership Function Layer**

Here, the input variable is fuzzified employing three Membership Functions (MFs), negative (N), zero (Z) and positive (P). Figure 7.2 shows these MFs where $N_v$ and $N_b$ respectively denote the right vertex and right base points of the MF N, $Z_{bl}$, $Z_{vl}$, $Z_{vr}$ and $Z_{br}$ respectively denote the left base, left vertex, right vertex and right base points of the MF Z and $P_b$ and $P_v$ respectively denote the left base and left vertex points of the MF P. The output of the *i*th MF is given as:

$$O_i^2 = \mu_i(u^1) = \mu_i(\Delta\hat{\mathbf{C}}_{Innk}(j,j)) \tag{7.31}$$

**3) Layer 3: Defuzzification layer**

This layer performs defuzzification where the defuzzified output is calculated as an weighted average of all its inputs. Hence the output from the solitary node in this layer can be calculated as:

$$O^3 = \frac{\sum_{i=1}^{3} O_i^2 * w_i}{\sum_{i=1}^{3} O_i^2} = \frac{\sum_{i=1}^{3} \mu_i(u^1) * w_i}{\sum_{i=1}^{3} \mu_i(u^1)} \tag{7.32}$$

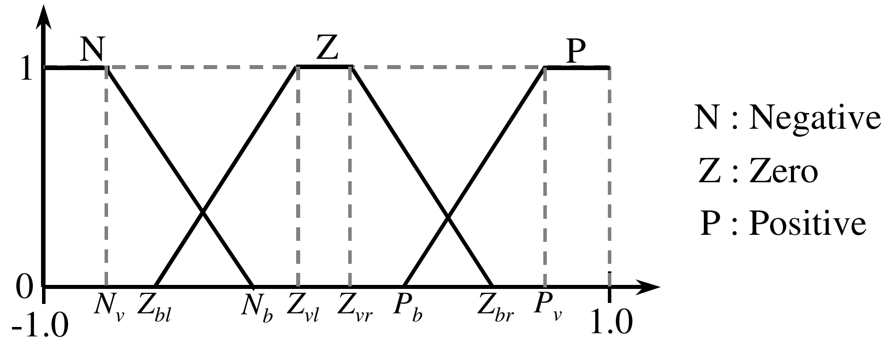


**Fig. 7.2.** Membership functions employed in Fig. 7.1. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).

**4) Layer 4: Output Layer**

This layer performs a suitable scaling for the defuzzified output. The input-output relationship of the node in this layer is given as:

$$O^4 = K * u^4 = K * O^3 \tag{7.33}$$

## *7.4.2 Training the Neuro-fuzzy Model Employing PSO*

This neuro-fuzzy model is trained to determine the suitable free parameters of the system i.e. the parameters of the MFs, the output consequence singletons and the output gain. However, the training cannot be accomplished in the conventional supervised mode, as the exact desired output, for a given input, is not quantitatively known. Hence, normal backpropagation kind of training methodology cannot be resorted to and it is suitable to apply stochastic global optimization algorithms for such systems in an unsupervised manner. There are several such candidate algorithms available now. In this section we describe how PSO can be suitably employed for this purpose. PSO is a relatively new algorithm [40], [41], that is based on the swarm behaviors of birds or fishes. The training of the neuro-fuzzy system is accomplished as a high-dimensional metaheuristic optimization problem, where the objective is to optimize a fitness function $f_{fit}(x_1, x_2, \cdots x_n)$ on the basis of the values of the variables $x_1, x_2, \cdots x_n$.

In a PSO problem, several such candidate solutions of $x_1, x_2, \cdots x_n$ are created in a multi-dimensional space (called "particles") and the suitability of each of them is evaluated in each iteration. For the problem under consideration here, each such potential "particle" is formed as a 12-dimensional vector $\mathbf{x}$ = $[x_1\ x_2 \cdots x_{12}]^T$, as shown in Fig. 7.3. Each "particle" $i$ is characterized by the vectors denoting its position ($\mathbf{x}_i$) and its velocity ($\mathbf{v}_i$) at the current time step. In order to pursue the optimum of the fitness function ($f_{fit}$), velocity $\mathbf{v}_i$ and hence position $\mathbf{x}_i$ of each particle is adjusted in each time step. The updated velocity in each time step $\mathbf{v}_{inew}$ is a function of three major components: the old velocity vector of the same particle ($\mathbf{v}_{iold}$), difference of the *i*th particle's best position found so far (called $\mathbf{p}_i$) and the current position ($\mathbf{x}_i$) (called the "cognitive" component) and difference of the best position of any particle within the context of the topological neighborhood of *i*th particle found so far (called $\mathbf{p}_g$) and current position of the ith particle ($\mathbf{x}_i$) (called the "social" component) [40, 41]. Each of the last two components is stochastically weighted so that the updating in the velocity of each particle will cause enough oscillations, allowing each particle to search for a better pattern within the problem space. Hence, the velocity and position update relations, in the *d*th dimension, are given as:

$$v_{idnew} = v_{idold} + \varphi_i(p_{id} - x_{id}) + \varphi_g(p_{gd} - x_{gd})$$

$$\textbf{IF } (v_{idnew} > v_{dmax}) \textbf{ THEN } v_{idnew} = v_{dmax} \textbf{ ENDIF}$$

$$\textbf{IF } (v_{idnew} < -v_{dmax}) \textbf{ THEN } v_{idnew} = -v_{dmax} \textbf{ ENDIF}$$

$$x_{idnew} = x_{idold} + v_{idnew}$$

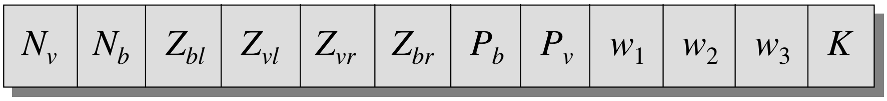
$$v_{idold} = v_{idnew}$$

$$x_{idold} = x_{idnew} \tag{7.34}$$

$\varphi_i$ and $\varphi_g$ are responsible for introducing stochastic weighting and they are given as $\varphi_i = c_i{*}rand_1(\ )$ and $\varphi_g = c_g{*}rand_2(\ )$. $rand_1(\ )$ and $rand_2(\ )$ are two random functions in [0, 1] and $c_i$ and $c_g$ are positive constants. A popular choice for $c_i$ and $c_g$ is $c_i = c_g = 2$. This traditional PSO model shows quick, aggressive convergence during the early phase but often encounters problem in fine tuning the search to determine the supreme solution. Hence, in our algorithm we have employed an improved version of this PSO algorithm that utilizes a judicious mix of aggressive, coarse updating during early iterations and fine updating during later iterations [40]. Hence the velocity update rule is given as

$$v_{\mathrm{idnew}} = w_{\mathrm{iter}}\,(v_{\mathrm{idold}}) + \varphi_{\mathrm{i}}(p_{\mathrm{id}} - x_{\mathrm{id}}) + \varphi_{\mathrm{g}}(p_{\mathrm{gd}} - x_{\mathrm{id}}) \tag{7.35}$$

with the position update rule remaining unchanged as given before. *w* is called the inertia weight which is initially kept high and then gradually decreased over the iterations so that it can initially introduce coarse adjustment in velocity updating and gradually fine changes in velocity updating takes over. In our algorithm, we have utilized linearly adaptable inertia weight and $w_{iter}$ gives the value of the inertia weight at that given iteration. The iterative process is continued until the optimization process yields a satisfactory result. This is evaluated on the basis of whether the value of $f_{fit}$ falls below the specified maximum allowable value or whether the maximum number of iterations has been reached. A detailed description of the PSO algorithm is available in [40, 41].



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $N_v$ | $N_b$ | $Z_{bl}$ | $Z_{vl}$ | $Z_{vr}$ | $Z_{br}$ | $P_b$ | $P_v$ | $w_1$ | $w_2$ | $w_3$ | $K$ |

**Fig. 7.3.** Detailed configuration of each 12-dimensional "particle" employed by PSO. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).

In our approach, the objective of the neuro-fuzzy assistance to the EKF based SLAM is to improve the estimation performance as much as possible. This means we should try and minimize the discrepancy between actual covariance and the theoretical covariance of the innovation sequence over the entire set of observation instants, during the movement of the vehicle/robot, as much as possible. Hence the fitness function is formulated on the basis of: *a*) computing the mean-square value of all the diagonal entries of the $\Delta\mathbf{C}_{Innk}$ matrix at any given observation instant, *b*) storing such mean-square values for each observation instant during an on-going iteration and *c*) computing a mean of all such mean-square values for all observation instants at the end of a complete iteration. Mathematically this can be shown as:

$$f_{fit} = \frac{\sum_{n_{obs}=1}^{N_{obs}} \left( \frac{\sum_{j=1}^{J_{C\_nobs}} [\Delta\mathbf{C}_{Innk}(j,j)]^2}{J_{C\_nobs}} \right)}{N_{obs}} \tag{7.36}$$

where $N_{obs}$ denotes the total number of observation instants in a given iteration and $J_{C\_nobs}$ denotes the total number of diagonal elements of $\Delta\mathbf{C}_{Innk}$ matrix when the $n_{obs}$th observation is made.

In the context of adapting a meaningful NFS, the positions of each "particle", at the end of each iteration, are subjected to several constraints. Most of these constraints are implemented to maintain specific shapes chosen for the MFs (usually trapezoidal, which as a special case can become triangular) and also to ensure that there is some overlapping between the stretches of consecutive MFs. Another constraint included is that, for each MF, its control points (starting from left to right) should be chosen in a monotonically nondecreasing fashion. This will ensure that all regions, within the universe of discourse of the input for the NFS, will remain covered by at least one MF. These constraints are implemented as:

```
IF ( $N_b < N_v$) THEN  $N_b = N_v$ ENDIF
IF ( $Z_{vl} < Z_{bl}$) THEN  $Z_{vl} = Z_{bl}$ ENDIF
IF ( $Z_{vr} < Z_{vl}$) THEN  $Z_{vr} = Z_{vl}$ ENDIF
IF ( $Z_{br} < Z_{vr}$) THEN  $Z_{br} = Z_{vr}$ ENDIF
IF ($P_v < P_b$) THEN  $P_v = P_b$ ENDIF
IF ( $N_b < Z_{bl}$) THEN  $N_b = Z_{bl}$ ENDIF
IF ( $Z_{br} < P_b$) THEN  $Z_{br} = P_b$ ENDIF                (7.37)
```

Another constraint is implemented to signify that the scaling employed in the output layer of the NFS is employed for magnitude scaling only, and hence it cannot be employed for changing polarity. It means that $K$ cannot become negative.

## 7.4.3 Performance Evaluation

To evaluate the performance of the proposed system, we have considered various environments, which are available in [42]. In fact the packages available in [42] should serve as an excellent platform for learning and analysis of existing Kalman filter and particle filter based SLAM algorithms. Researchers can develop their own algorithms and can compare their performance vis-à-vis these algorithms. Several benchmark environments are available there and we have tested our algorithm in these simulated environments with their associated given vehicle motion model. The environment is usually specified in such a manner where a vehicle/robot is supposed to navigate through some waypoints and in the process should be able to acquire the map of the environment with several configurations of feature/landmark points. In the present scheme, we consider three such environments as specified in [42]. In each case we have the identical scene of ideal robot movement where the robot path is specified by 17 waypoints. However, each environment consists of varied number of landmarks to impose several degrees of complexities and the three environments under consideration consist of 35, 135 and 497 landmarks respectively. The uncertainties in control inputs are specified as: $\sigma_w$ = 0.3 m/sec. and $\sigma_s$ = 3 deg. An observation step and the associated update step is carried out after eight consecutive prediction steps, identical with the EKF based algorithm in [42]. This follows a popular notion in EKF-based SLAM community, where instead of employing an observation and update step after each prediction step, one computes several consecutive prediction steps, and then takes corrective action by one observation and update step. This helps in reducing the computational burden of the SLAM algorithm. In algo. 7.1, this is indicated by the *Time_for_Observation* flag, which is set TRUE for one iteration, after each 8 successive iterations.

The performance of the proposed system is compared with a conventional EKF-based SLAM system where the **Q** and **R** matrices are kept static throughout the experiment. The proposed algorithm starts with the same **Q** and **R** matrices, but it keeps adapting the **R** matrix according to the proposed scheme. According to the data available from [42], the EKF based algorithm works perfectly when sensor statistics are known as: $\sigma_r = 0.1$ m. and $\sigma_\theta = 1$ deg. First we consider the situation where the sensor statistics are wrongly considered as: $\sigma_r = 2.0$ m. and $\sigma_\theta = 0.1$ deg. In each figure, the firm lines shown in green, depict the actual path traversed by the robot, while the firm lines shown in black, depict the SLAM estimated path traversed based on estimated states of robot poses in each sampling instant or iteration. The stars (∗) depict the actual landmark positions, which are stationary in the environment. The crosses (+) depict the positions of these landmarks estimated at the end of the test run. Obviously, the performance of the system will be superior, if the estimated robot path and actual path match as far as possible and the estimated landmark positions and their actual positions coincide as far as possible. Figure 7.4(a) to Fig. 7.4(c) shows the performance of the conventional EKF-based SLAM for three different environment situations. It can be seen that the performance is acceptable when there are small number of landmarks in the environment. However, the performance became really bad when the landmarks became denser and both the estimations of the robot pose at different instants and the map acquired degraded significantly as the EKF estimations are quite distant from the original robot positions and the map situation. Figure 7.5(a) to Fig. 7.5(c) show the situations when the neuro-fuzzy assisted EKF-based SLAM is employed for identical environments. It can be seen that the neuro-fuzzy assistance could improve the situation dramatically and the estimates of the robot states as well as acquisition of the map was quite stable for all three different environments with varied number of landmarks. In all these environments, the robot position estimates follow the actual robot positions closely and the estimation of the stationary landmark positions also closely matches with their actual positions in the environments.

The scheme was further tested for another situation where the sensor statistics are wrongly considered in opposite directions and they are considered as $\sigma_r = 0.01$ m. and $\sigma_\theta = 3.0$ deg. Then the same set of algorithms was employed for identical set of environments. Figure 7.6(a) to Fig. 7.6(c) show the performances of the conventional EKF-based SLAM and Fig. 7.7(a) to Fig. 7.7(c) show the corresponding performances of the neuro-fuzzy assisted EKF-based SLAM algorithms. In these case studies, the EKF-based SLAM shows a different trend in performance. As we can see, the estimation performance is worst for the environment containing small number of landmarks. However, with increase in landmarks, the estimations became more accurate and for the situation with 497 landmarks, the performance of the EKF-based SLAM was quite satisfactory. On the other hand, the neuro-fuzzy assisted EKF showed uniformly stable performance for each environment with quite accurate estimations of robot poses and feature positions for each environment situation. Each result, shown in Fig. 7.5(a) to Fig. 7.5(c) and Fig. 7.7(a) to Fig. 7.7(c), for the neuro-fuzzy assisted EKF based SLAM depicts one sample run conducted. For each of these six specific situations of two case studies, we conducted 10 individual runs. It was found that, for each given situation, results obtained with each of 10 individual runs, were very close to each other. These case studies further prove that the neuro-fuzzy assistance can vastly improve the degrading performance of the traditional EKF algorithm in several situations, when the sensor statistics are wrongly known. In these situations, the performance of the conventional EKF becomes highly unreliable. However, presence of neuro-fuzzy assistance can help the EKF to maintain a stable performance and this performance has been shown robust enough over several environment situations, with several wrong knowledge of sensor statistics.

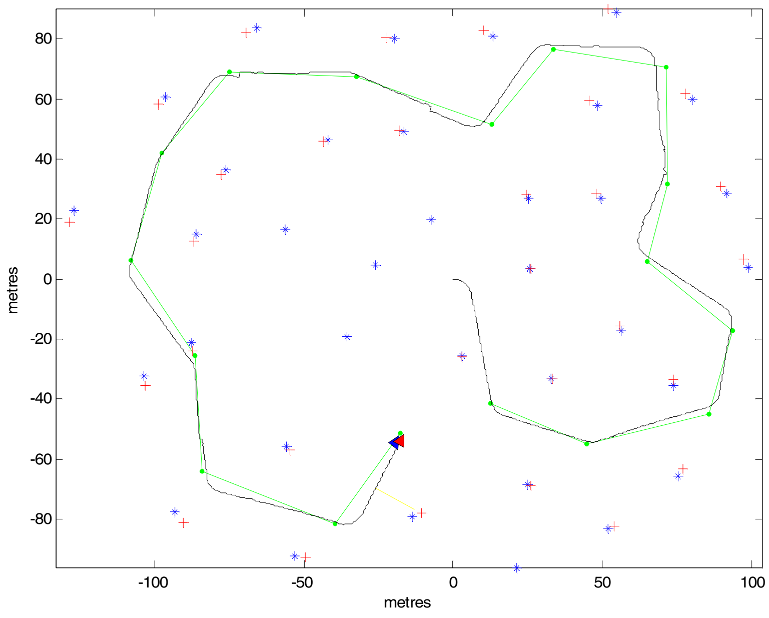
For the neuro-fuzzy assisted EKF based SLAM, the training of the neuro-fuzzy system, for each case study as described before, was carried out in offline situation on the basis of the data gathered by the robot for a given environment situation. For our experimentation, we implemented the training procedure, for each case study, for the environment containing 135 landmarks. Once the training of the neuro-fuzzy system was completed (on the basis of a given configuration of the landmarks) and the free parameters of the NFS were suitably determined, the trained NFS-based EKF was implemented for robot navigation through the waypoints for several configuration of landmarks as described before (i.e. environments with 35, 135 and 497 landmarks). Table 7.1 details these parameters employed for the PSO algorithm employed for training the NFS. Here, the dimensions of each particle, which are employed to learn the control points of the MFs of the NFS (i.e. $[x_1\ x_2 \cdots x_8]$), are all initialized with their positions within the range [-1, 1]. This is done in conformation with the normalization procedure that works in conjunction with the NFS. The prospective weights associated with the layer 3 of the NFS (denoted by the dimensions $x_9$, $x_{10}$ and $x_{11}$ of the PSO algorithm) are all initialized with their positions within the range [-2, 2]. The prospective gain $K$ associated with the layer 4 of the NFS (denoted by the dimension $x_{12}$ of the PSO algorithm) is initialized with its position within the range [0, 2], because it is assumed that $K$ is a non-negative quantity. Each time, the termination criterion for the PSO algorithm was set for a maximum number of iterations (*maxiter*) of 20. For the case study with initial sensor information $\sigma_r = 2.0$ m. and $\sigma_\theta = 0.1$ deg, the learned parameters of the NFS at the completion of the training procedure are:

$[x_1\ x_2 \cdots x_{12}]$ = [-0.2008 –0.0626 –0.0626 –0.0626 0.0820 0.5961 0.3224 0.4002 –0.0086 1.5801 –0.9729 0.0011]

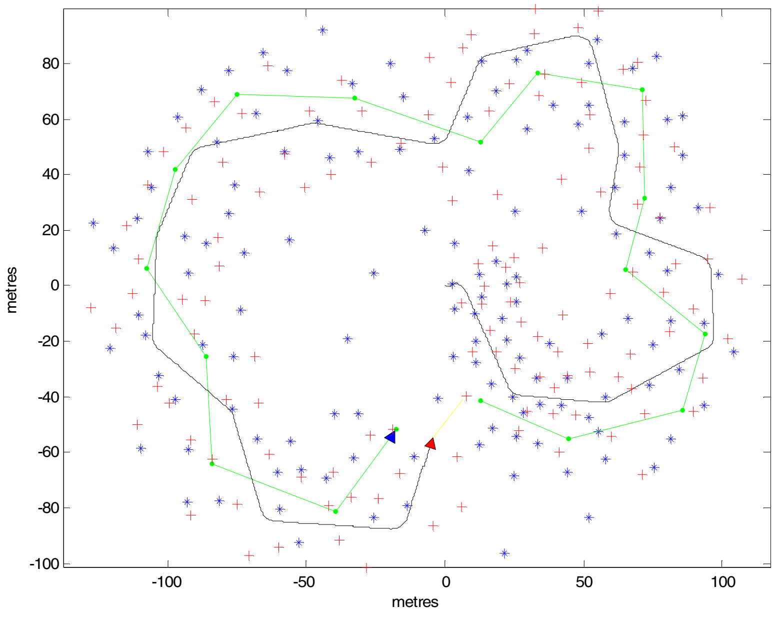
and for the case study with initial sensor information $\sigma_r = 0.01$ m. and $\sigma_\theta = 3.0$ deg., the learned parameters of the NFS are:

$[x_1\ x_2 \cdots x_{12}]$ = [-0.4570 0.5242 0.4805 0.9741 0.9741 0.9741 –0.4290 0.2413 –0.0024 –0.8762 1.3561 0.2907].

In each case, it can be seen that these learned parameters satisfied those constraints presented in (7.37).

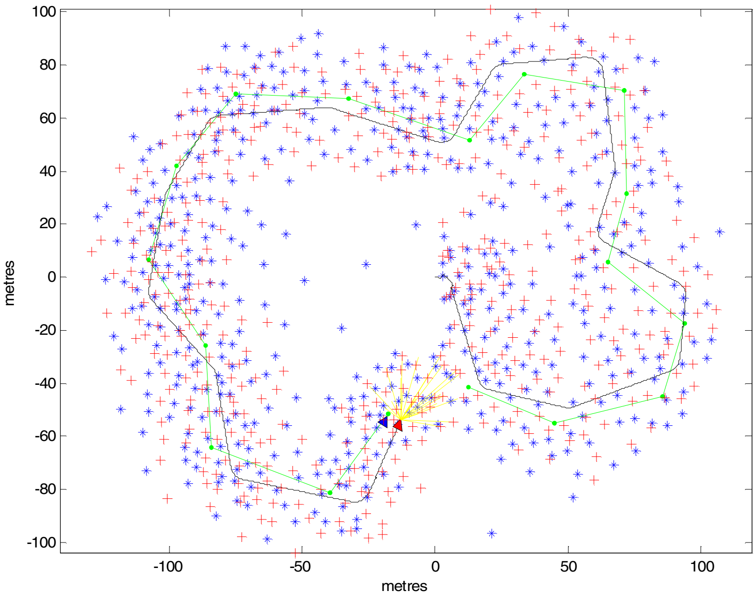


(a)



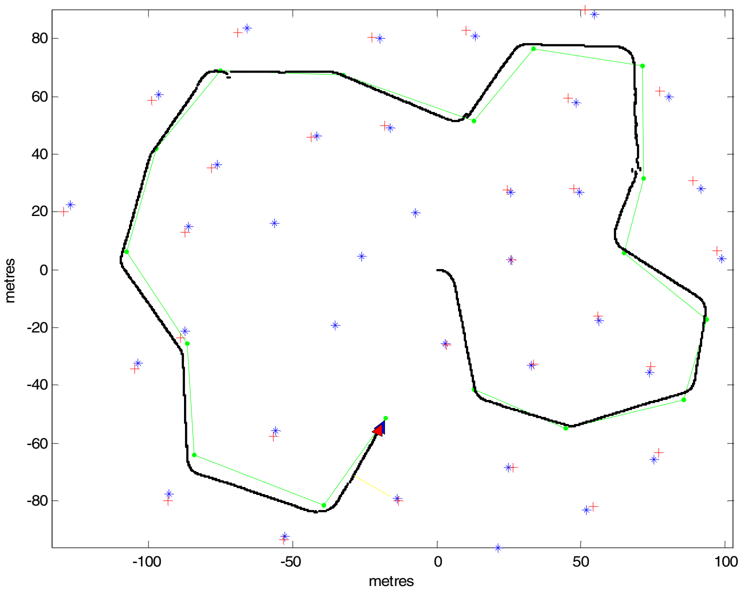
(b)

**Fig. 7.4.** Conventional EKF-based SLAM performance for case study I ($\sigma_r = 2.0$ m. and $\sigma_b = 0.1$ deg.) with (a) 35, (b) 135 and (c) 497 features/landmarks in the environment. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).



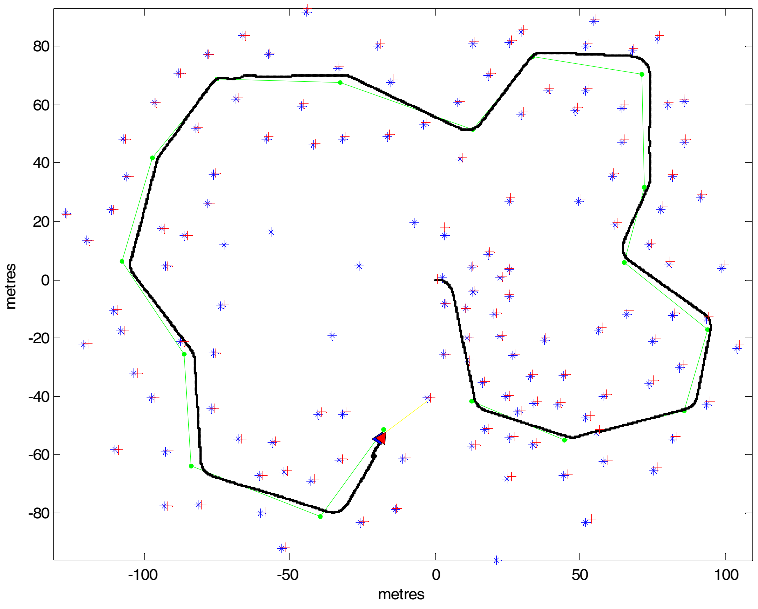
(c)

**Fig. 7.4.** (*continued*)

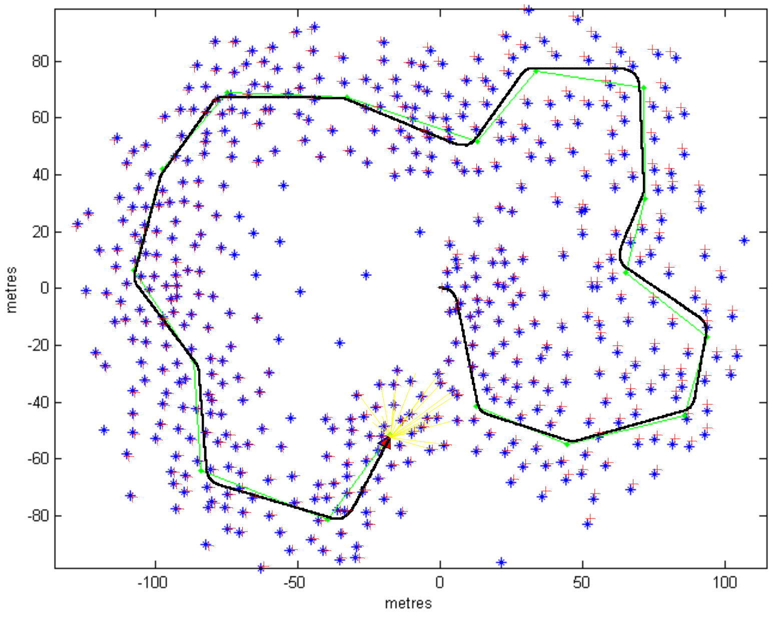


(a)

**Fig. 7.5.** Neuro-fuzzy assisted EKF-based SLAM performance for case study I ($\sigma_r = 2.0$ m. and $\sigma_b = 0.1$ deg.) with (a) 35, (b) 135 and (c) 497 features/landmarks in the environment. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).

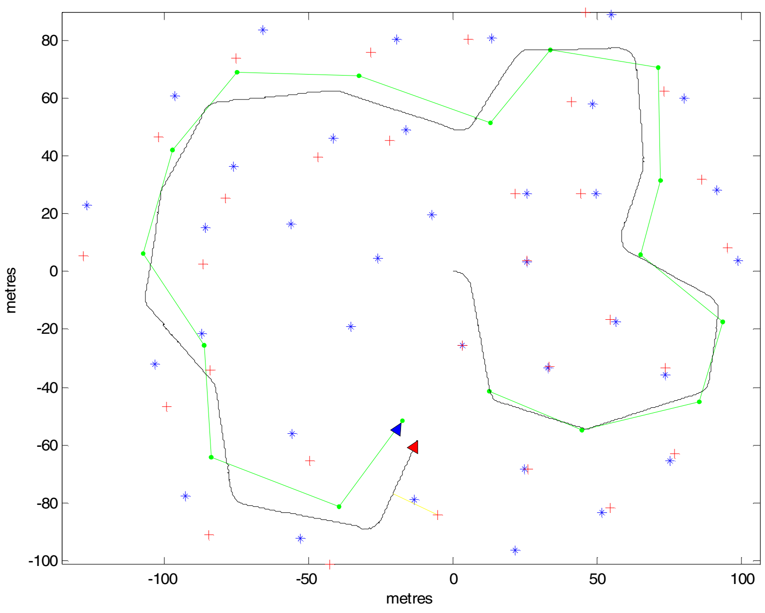


(b)

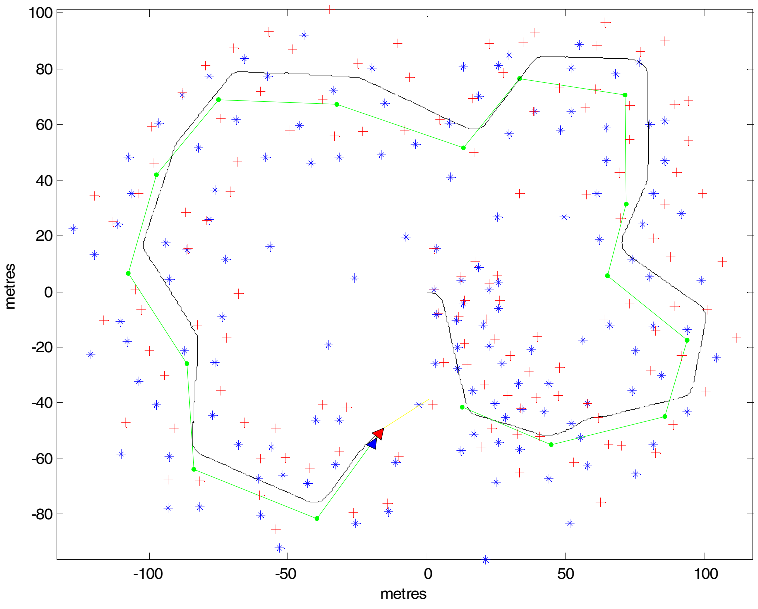


(c)

**Fig. 7.5.** (*continued*)

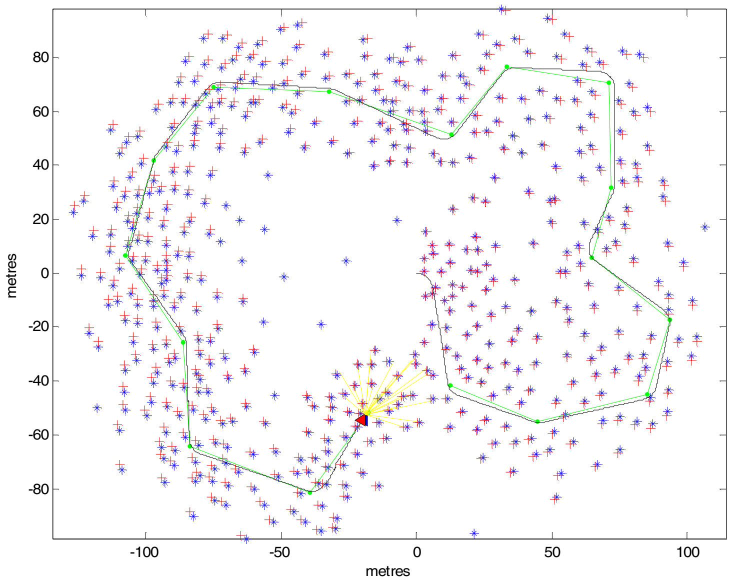


(a)



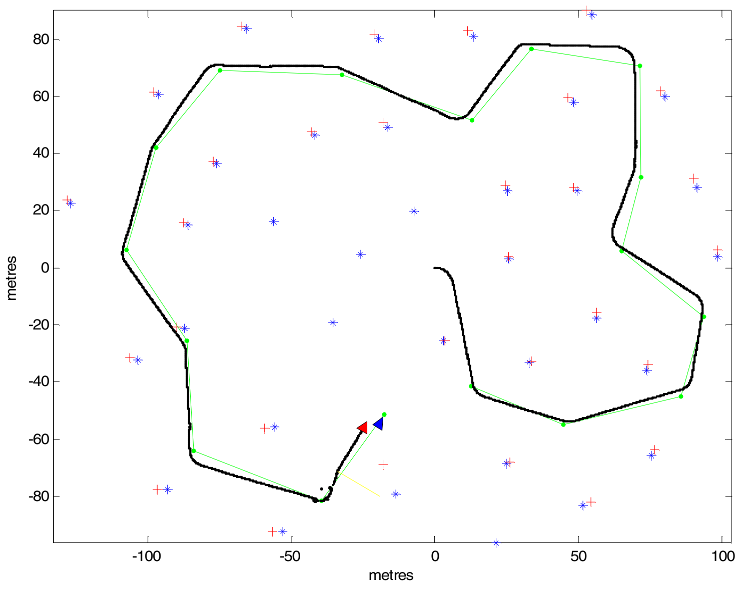
(b)

**Fig. 7.6.** Conventional EKF-based SLAM performance for case study II ($\sigma_r = 0.01$ m. and $\sigma_b = 3.0$ deg.) with (a) 35, (b) 135 and (c) 497 features/landmarks in the environment. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).



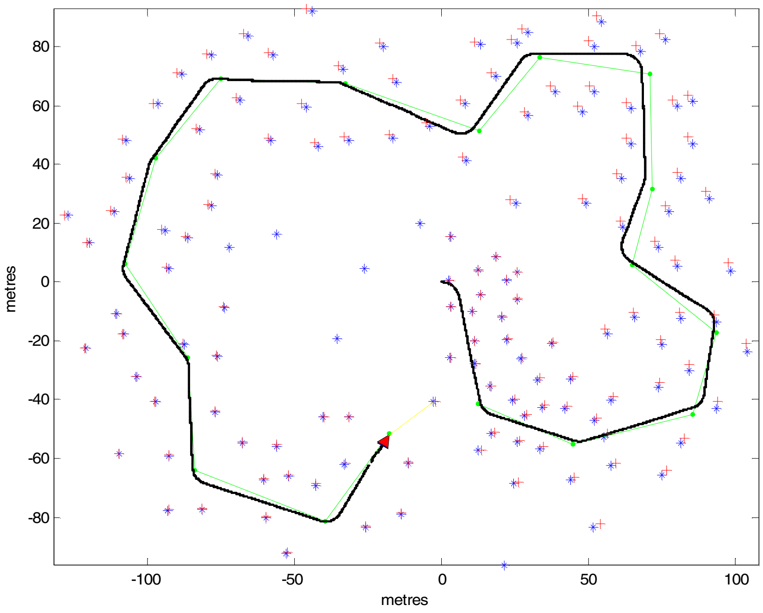
(c)

**Fig. 7.6.** (*continued*)

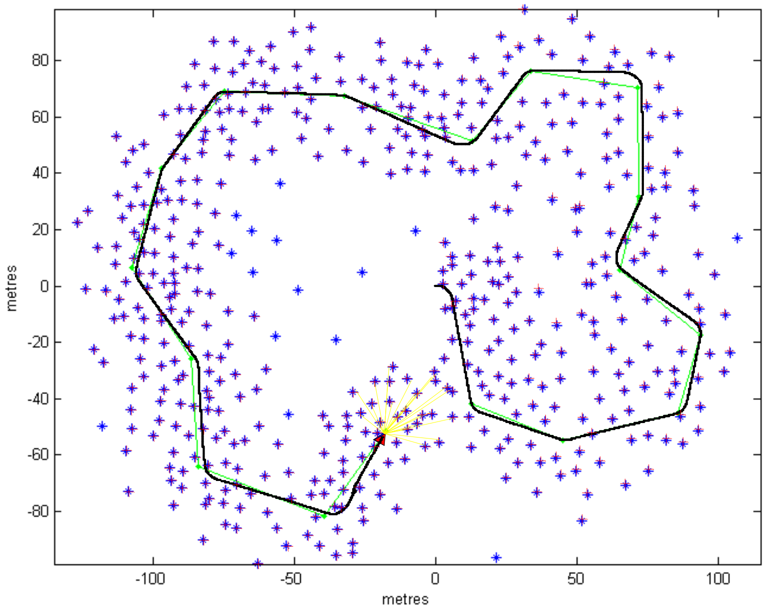


(a)

**Fig. 7.7.** Neuro-fuzzy assisted EKF-based SLAM performance for case study II ($\sigma_r = 0.01$ m. and $\sigma_b = 3.0$ deg.) with (a) 35, (b) 135 and (c) 497 features/landmarks in the environment. (Reproduced from [44] with permission from the IEEE. ©2007 IEEE.).



(b)



(c)

**Fig. 7.7.** (*continued*)

**Table 7.1.** The PSO parameters employed

| Sl. No. | Parameter descriptions | Parameter values for case study (*i*) | Parameter values for case study (*ii*) |
|---|---|---|---|
| 1 | No. of particles ($N$) | 40 | 40 |
| 2 | No. of dimensions ($D$) | 12 | 12 |
| 3 | Initial inertia weight ($W_{\text{initial}}$) | 0.9 | 0.9 |
| 4 | Slope of inertia weight ($\Delta W$) | 2.5e-4 | 2.5e-4 |
| 5 | Initialization range for MFs ($x_1$, $x_2$, … $x_8$) | [-1, 1] | [-1, 1] |
| 6 | Initialization range for weight factors ($x_9$, $x_{10}$, $x_{11}$) | [-2, 2] | [-2, 2] |
| 7 | Initialization range for gain ($x_{12}$) | [0, 2] | [0, 2] |
| 8 | Maximum permissible velocity for MFs ($v_{1\max}$, $v_{2\max}$, … $v_{8\max}$) | 0.3 | 0.1 |
| 9 | Maximum permissible velocity for weight factors ($v_{9\max}$, $v_{10\max}$, $v_{11\max}$) | 1.0 | 0.5 |
| 10 | Maximum permissible velocity for gain ($v_{12\max}$) | 1.0 | 0.5 |

## 7.5 Training a Fuzzy Supervisor Employing Differential Evolution (DE) Based Optimization

In the previous section we demonstrated how PSO can be utilized to train a fuzzy/neuro-fuzzy supervisor for successful supervision of an EKF based SLAM system. Logically speaking, the idea can be extended to employ other evolutionary algorithms too for similar fuzzy/neuro-fuzzy based supervision purpose. Hence we implemented a similar fuzzy supervisor employing differential evolution (DE), another popular evolutionary algorithm known, for similar types of problems [45]. In DE, 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 multi-dimensional 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 $(x_1, x_2, \cdots x_D)$ 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 *i*th 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 *G*th generation. This can be shown by the following formula [46],[47]:

$$v_{i,G+1} = x_{r_1,G} + F\,(x_{r_2,G} - x_{r_3,G}) \tag{7.38}$$

where $i = 1,2,\cdots,NP$ . 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]:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \cdots u_{Di,G+1}) \text{ with}$$

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{for} \quad j = \langle n+1 \rangle_D \cdots \langle n+L \rangle_D \\ x_{ji,G} & \text{for all other} \quad j \in [1, D] \end{cases} \tag{7.39}$$

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 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 $\boldsymbol{x}_{i,G+1} = \boldsymbol{x}_{i,G}$.

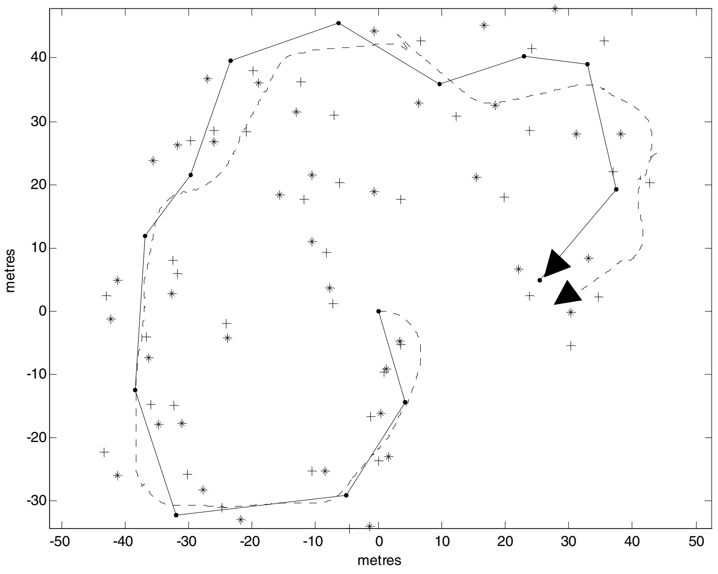
## 7.5.1 Performance Evaluation

The performance of DE optimized fuzzy supervisor based solution for the SLAM problems has also been tested by creating an environment in simulation, utilizing the package available in [42], as done in our previous set of case studies. For the new set of case studies, we consider a different environment and 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. For these situations, the performances exhibited by the conventional EKF-based SLAM [42] are shown in Fig. 7.8(a) and Fig. 7.8(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 our DE-optimized fuzzy supervisor based 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. 7.9(a) and Fig. 7.9(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 differential evolution 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 [46], [47]. 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 [48]. 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.

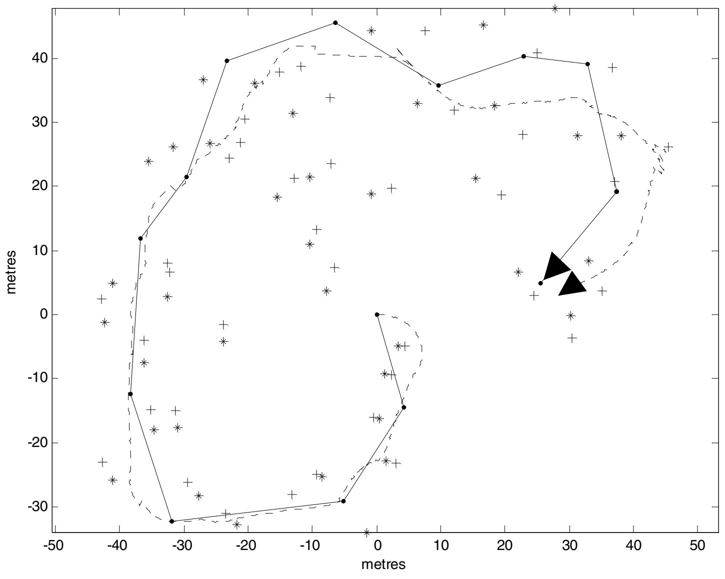
The results shown in Fig. 7.9 are obtained in the implementation phase, using the fuzzy supervisors trained by the DE algorithm, with the chosen control parameters $NP = 20$, $F = 0.1$, $CR = 0.5$. Like most other stochastic global optimization methods, the performance of the differential evolution strategy 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 [46], for many applications, choices of $NP = 10{*}D$, $F \in [0.5, 1]$ and $CR \in [0, 1]$ but much lower than 1, are considered to be good choices. Among these factors, $F$ is considered to 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 varied $F$ for a number of values in the range 0 to 1 and for each case the fuzzy supervisor was trained separately. Although, according to the general guideline $NP$ should have been chosen as 10*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 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 7.10(a) to Fig. 7.10(c) show the corresponding performances of the system in the implementation phase with the trained fuzzy supervision for $F = 0.05$, $F = 0.1$ and $F = 0.5$. Figure 7.11 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 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.

With this value of $F$, then 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 7.12 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 7.13 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. (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.). Figure 7.9(a) and Fig. 7.9(b) showed the performances of those case studies, in the implementation phase.

In the next phase, we present a performance comparison between the fuzzy supervisor tuned by DE and the fuzzy supervisor tuned by 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.). The popular version of PSO, employed using linearly decreasing inertia weight, as described in (7.35), is used for this purpose. 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 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, $W_{\text{initial}}$, 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 7.14 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 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 7.15 shows a similar comparison of estimation performance for the best PSO-tuned and best DE-tuned fuzzy supervisors for the

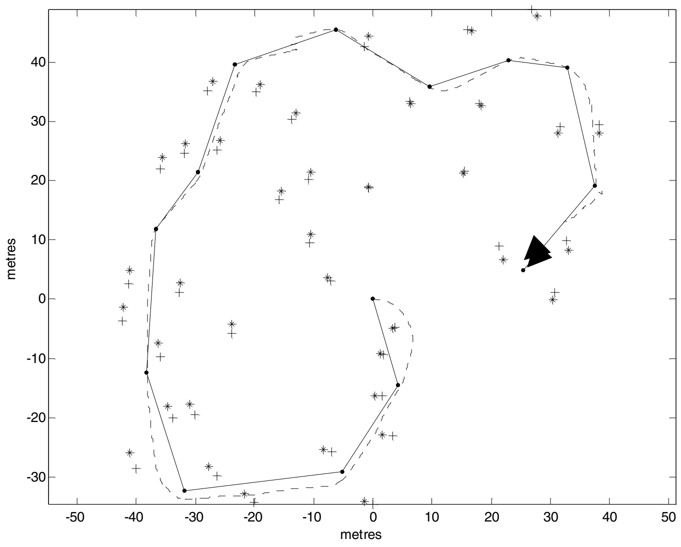


(a)

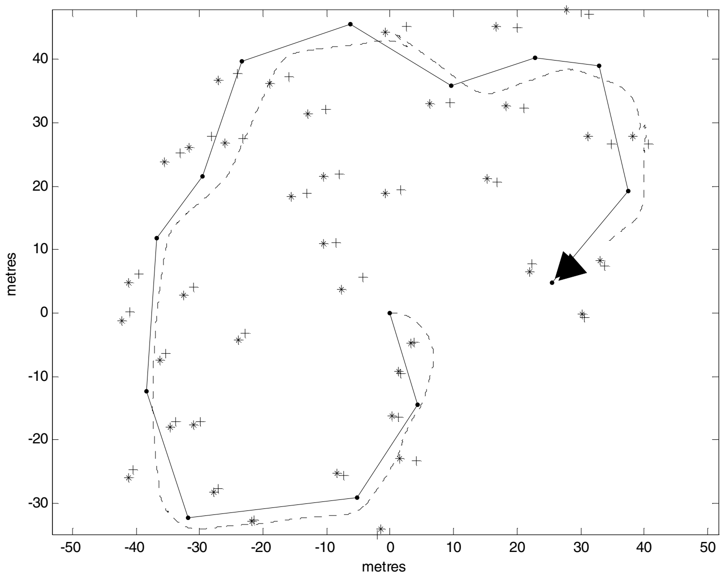


(b)

**Fig. 7.8.** 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.)

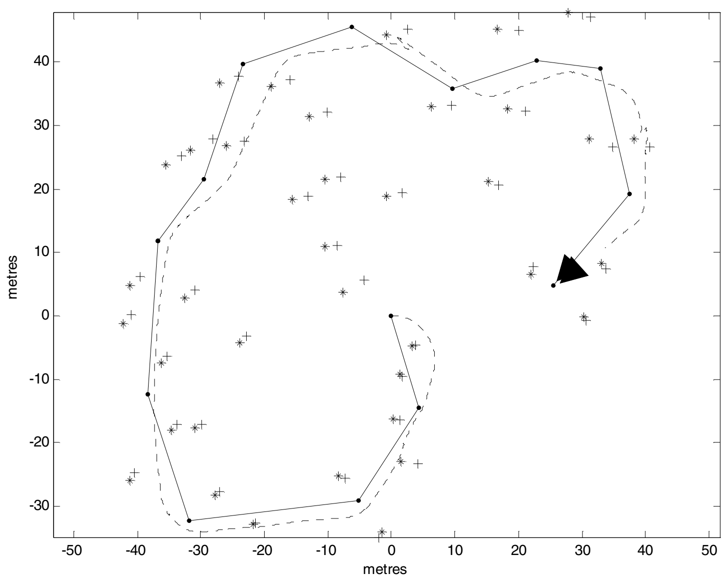


(a)

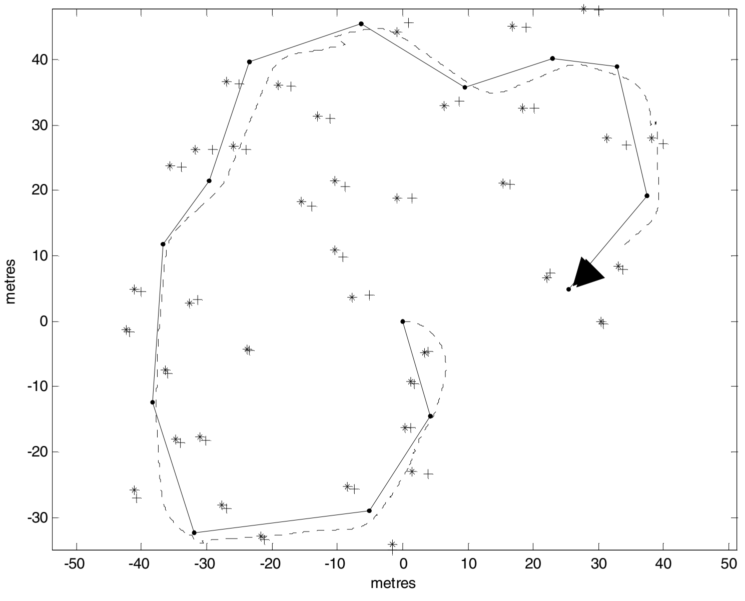


(b)

**Fig. 7.9.** 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.)

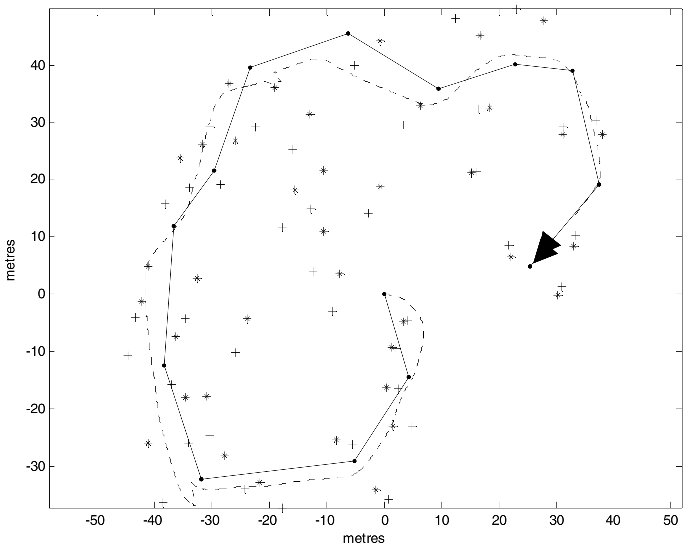


(a)



(b)

**Fig. 7.10.** 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



(c)

**Fig. 7.10.** (*continued*)

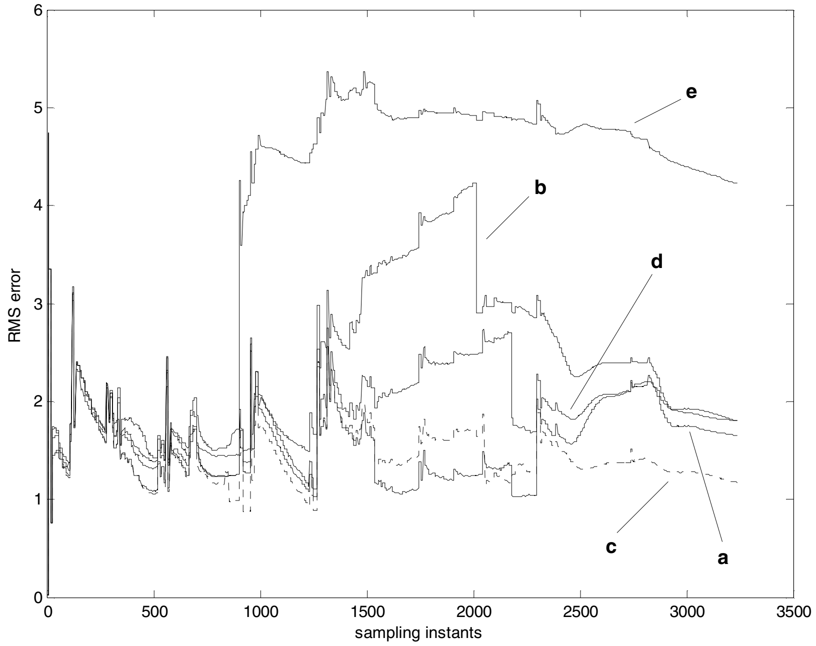


**Fig. 7.11.** The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the DE-based training was carried out with $NP = 20$, $CR = 0.5$, and (a) $F = 0.05$, (b) $F = 0.08$, (c) $F = 0.1$, (d) $F = 0.15$, and (c) $F = 0.5$

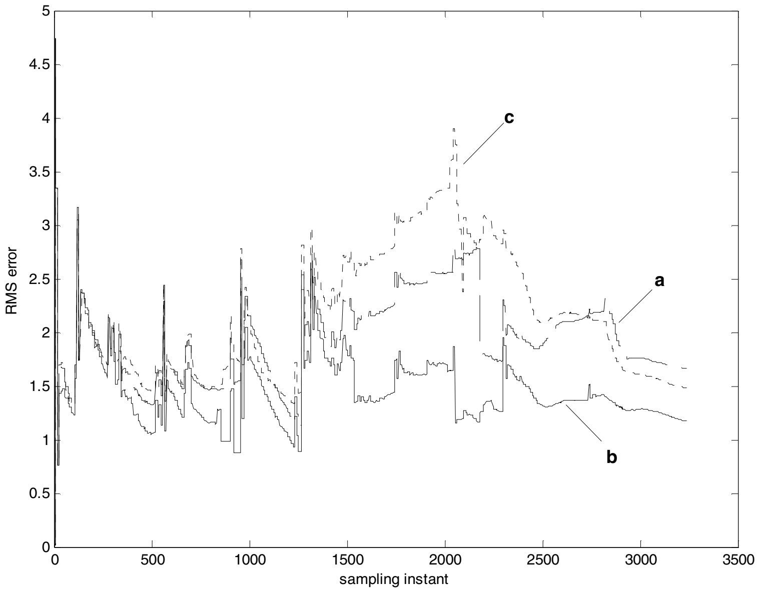


**Fig. 7.12.** The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the DE-based training was carried out with $F = 0.1$, $CR = 0.5$, and (a) $NP = 15$, (b) $NP = 20$, and (c) $NP = 25$

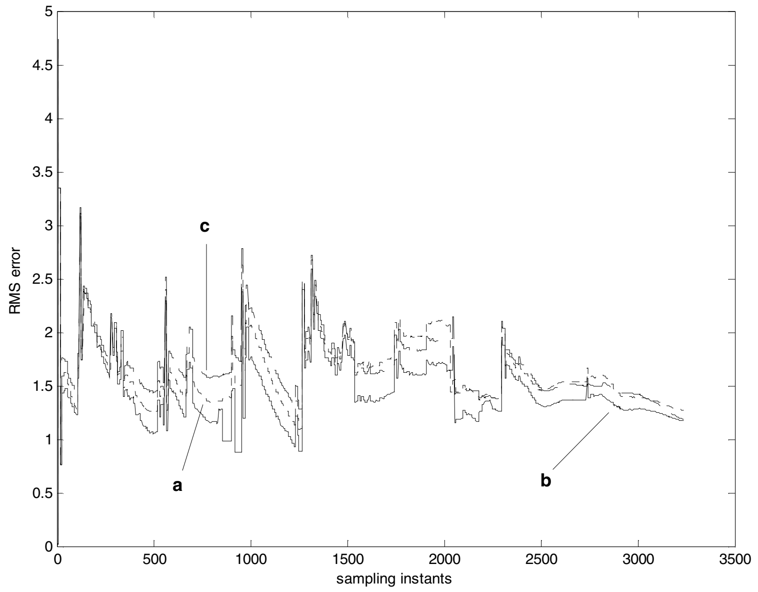


**Fig. 7.13.** The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the DE-based training was carried out with $F = 0.1$, $NP = 20$, and (a) $CR = 0.4$, (b) $CR = 0.5$, and (c) $CR = 0.6$

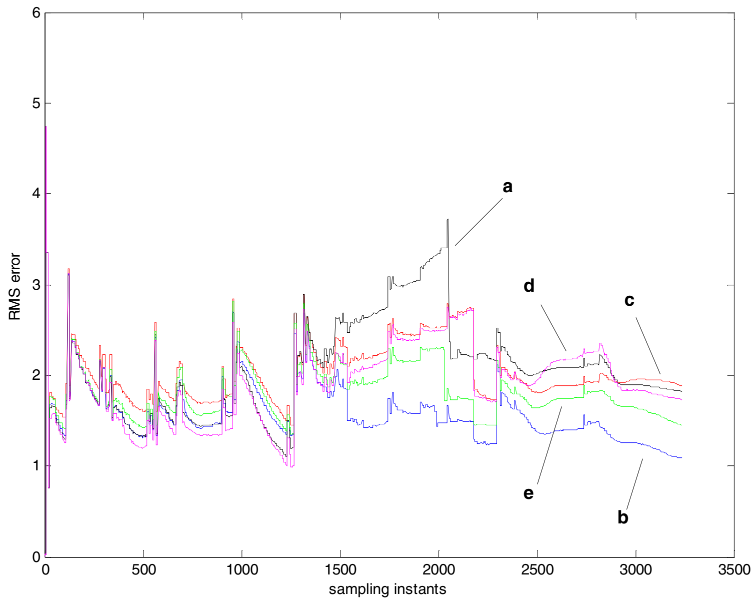


**Fig. 7.14.** The estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the PSO-based training was carried out with the slope of inertia weight chosen as (a) 2.0e-4, (b) 2.5e-4, (c) 5.0e-4, (d) 4e-2, and (e) 5e-2

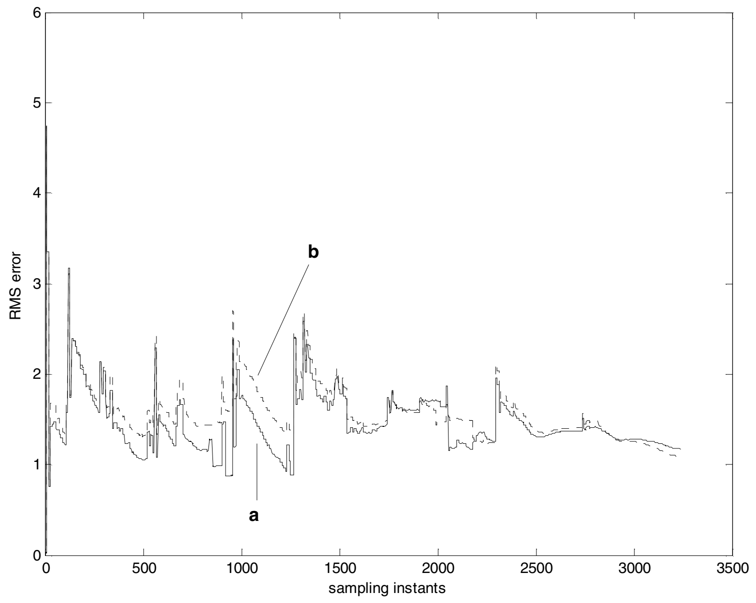


**Fig. 7.15.** Comparison of the estimation performance of the fuzzy supervised EKF-based SLAM, in the implementation phase, when the fuzzy supervisor is trained by (a) DE algorithm and (b) PSO algorithm

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. However we would like to generally remark that this performance may vary depending on the environment chosen and the sensor statistics considered.

## 7.6 Summary

The present chapter discussed the importance of SLAM in the context of mobile robot navigation and, at first, described the extended Kalman filter based SLAM algorithms in detail. Next we considered the degradation in system performance when *a priori* knowledge of the sensor statistics is incorrect and showed how fuzzy/neuro-fuzzy assistance or supervision can significantly improve the performance of the algorithm. Usually, EKF is known as a good choice for SLAM algorithms when the associated statistical models are well known. However, the performance can become significantly unpredictable and degrading when the knowledge of such statistics is inappropriate. The fuzzy/neuro-fuzzy supervisor based system proposes to start the system with the wrongly known statistics and then adapt the **R** matrix, online, on the basis of a fuzzy/neuro-fuzzy system that attempts to minimize the mismatch between the theoretical and the actual values of the innovation sequence. The free parameters of the neuro-fuzzy system are automatically learned employing an evolutionary optimization based training procedure. The chapter showed how two popular contemporary evolutionary optimization techniques, namely, PSO and DE, can be utilized successfully for this purpose. The performance evaluation is carried out for several benchmark environment situations with several wrong knowledge of sensor statistics. While the conventional EKF based SLAM showed unreliable performance with significant degradation in many situations, the fuzzy/neuro-fuzzy assistance could improve this EKF's performance significantly and could provide robust, accurate performance in each sample situation in each case study.

**Acknowledgement.** This work was partially supported by JSPS Postdoctoral Fellowship for Foreign Researchers in Japan. This work was also partially supported by All India Council for Technical Education under RPS scheme (Grant No. 8023/BOR/RPS-89/2006-07).

## References

[1] Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F.: A solution to the simultaneous localization and map building (SLAM) problem. IEEE Tran. Robotics and Automation 17(3), 229–241 (2001)

[2] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico (2003)

[3] Grisetti, G., Stachniss, C., Burgard, W.: Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA), Barcelona, Spain, pp. 2443–2448 (2005)

[4] Smith, R., Cheeseman, P.: On the representation and estimation of spatial uncertainty. International Journal of Robotics Research 5(4) (1986)

[5] Moutarlier, P., Chatila, R.: Stochastic multisensory data fusion for mobile robot location and environment modeling. In: 5th Int. Symposium on Robotics Research, Tokyo (1989)

[6] Davison, A.J.: Mobile Robot Navigation Using Active Vision. PhD Thesis, Univ. of Oxford (1998)

[7] Bailey, T.: Mobile Robot Localization and Mapping in Extensive Outdoor Environments. PhD Thesis, Univ. of Sydney (2002)

[8] Davison, A.J., Murray, D.W.: Simultaneous localization and map-building using active vision. IEEE Tran. Pattern Analysis and Machine Intelligence 24(7), 865–880 (2002)

[9] Guivant, J., Nebot, E.: Optimization of the simultaneous localization and map-building algorithm and real-time implementation. IEEE Tran. Robotics and Automation 17(3), 242–257 (2001)

[10] Guivant, J., Nebot, E.: Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. IEEE Tran. Robotics and Automation 19(4), 749–755 (2003)

[11] Williams, S.B., Newman, P., Dissanayake, G., Durrant-Whyte, H.: Autonomous underwater simultaneous localization and map building. In: Proc. IEEE International Conference on Robotics and Automation, San Francisco, CA, vol. 2, pp. 1792–1798 (2000)

[12] Chong, K.S., Kleeman, L.: Feature-based mapping in real, large scale environments using an ultrasonic array. International Journal of Robotic Research 18(2), 3–19 (1999)

[13] Bosse, M., Leonard, J., Teller, S.: Large-scale CML using a network of multiple local maps. In: Leonard, J., Tardós, J.D., Thrun, S., Choset, H. (eds.) Workshop Notes of the ICRA Workshopon Concurrent Mapping and Localization for Autonomous Mobile Robots (W4), Washington, DC. ICRA Conference (2002)

[14] Thrun, S., Fox, D., Burgard, W.: A probabilistic approach to concurrent mapping and localization for mobile robots. Machine Learning 31, 29–53 (1998); also appeared in Autonomous Robots 5, 253–271 (joint issue)

[15] Williams, S., Dissanayake, G., Durrant-Whyte, H.F.: Towards terrain-aided navigation for underwater robotics. Advanced Robotics 15(5) (2001)

[16] Thrun, S., Hähnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., Whittaker, W.: A system for volumetric robotic mapping of abandoned mines. In: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA (2003)

[17] Castellanos, J.A., Montiel, J.M.M., Neira, J., Tardós, J.D.: The SPmap: A probabilistic framework for simultaneous localization and map building. IEEE Transactions on Robotics and Automation 15(5), 948–953 (1999)

[18] Paskin, M.A.: Thin junction tree filters for simultaneous localization and mapping. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico (2003)

[19] Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., Ng, A.Y.: Simultaneous mapping and localization with sparse extended information filters. In: Boissonnat, J.-D., Burdick, J., Goldberg, K., Hutchinson, S. (eds.) Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics, Nice, France (2002)

[20] Neira, J., Tardós, J.D.: Data association in stochastic mapping using the joint compatibility test. IEEE Transactions on Robotics and Automation 17(6), 890–897 (2001)

[21] Shatkay, H., Kaelbling, L.: Learning topological maps with weak local odometric information. In: Proceedings of IJCAI 1997. IJCAI, Inc. (1997)

[22] Araneda, A.: Statistical inference in mapping and localization for a mobile robot. In: Bernardo, J.M., Bayarri, M.J., Berger, J.O., Dawid, A.P., Heckerman, D., Smith, A.F.M., West, M. (eds.) Bayesian Statistics 7. Oxford University Press, Oxford (2003)

[23] Montemerlo, M., Thrun, S.: Simultaneous localization and mapping with unknown data association using Fast SLAM. In: Proc. IEEE International Conference on Robotics and Automation (ICRA), Taipei, Taiwan (2003)

[24] Hu, W., Downs, T., Wyeth, G., Milford, M., Prasser, D.: A modified particle filter for simultaneous robot localization and Landmark tracking in an indoor environment. In: Proc. Australian Conference on Robotics and Automation (ACRA), Canberra, Australia (2004)

[25] Frese, U., Larsson, P., Duckett, T.: A multilevel relaxation algorithm for simultaneous localization and mapping. IEEE Tran. Robotics 21(2), 196–207 (2005)

[26] Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: A factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, AAAI (2002)

[27] Lu, F., Milios, E.: Globally consistent range scan alignment for environment mapping. Autonomous Robots 4, 333–349 (1997)

[28] Mehra, R.K.: On the identification of variances and adaptive Kalman filtering. IEEE Tran. Automatic Control AC-15(2), 175–184 (1970)

[29] Fitzgerald, R.J.: Divergence of the Kalman filter. IEEE Tran. Automatic Control AC-16(6), 736–747 (1971)

[30] Sinha, N.K., Tom, A.: Adaptive state estimation for systems with unknown noise covariances. International Journal of Systems Science 8(4), 377–384 (1977)

[31] Bellanger, P.R.: Estimation of noise covariance matrices for a linear time-varying stochastic process. Automatica 10, 267–275 (1974)

[32] Dee, D.P., Cohn, S.E., Dalcher, A., Ghil, M.: An efficient algorithm for estimating noise covariances in distributed systems. IEEE Tran. Automatic Control AC-30(11), 1057–1065 (1985)

[33] Reynolds, R.G.: Robust estimation of covariance matrices. IEEE Tran. Automatic Control 32(9), 1047–1051 (1990)

[34] Morikawa, H., Fujisaki, H.: System identification of the speech production process based on a state-space representation. IEEE Trans. Acoust., Speech, Signal Processing ASSP-32, 252–262 (1984)

[35] Noriega, G., Pasupathy, S.: Adaptive estimation of noise covariance matrices in real-time preprocessing of geophysical data. IEEE Trans. Geoscience and Remote Sensing 35(5), 1146–1159 (1997)

[36] Kobayashi, K., Cheok, K.C., Watanabe, K., Munekata, F.: Accurate differential global positioning system via fuzzy logic Kalman filter sensor fusion technique. IEEE Tran. Industrial Electronics 45(3), 510–518 (1998)

[37] Loebis, D., Sutton, R., Chudley, J., Naeem, W.: Adaptive tuning of a Kalman filter via fuzzy logic for an intelligent AUV navigation system. Control Engineering Practice 12, 1531–1539 (2004)

[38] Wu, Z.Q., Harris, C.J.: An adaptive neurofuzzy Kalman filter. In: Proc. 5th International Conference on Fuzzy Sets and Systems FUZZ-IEEE 1996, vol. 2, pp. 1344–1350 (September 1996)

[39] Sasiadek, J.Z., Wang, Q., Zeremba, M.B.: Fuzzy adaptive Kalman filtering for INS/GPS data fusion. In: Proc. 15th International Symposium on Intelligent Control (ISIC 2000), Rio, Patras, Greece (July 2000)

[40] Clerc, M., Kennedy, J.: The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Tran. Evolutionary Computation 6(1), 58–73 (2002)

[41] Shi, Y., Eberhart, R.C.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congr. Evolutionary Computation, pp. 1945–1950. IEEE Service Center, Piscataway (1999)

[42] `http://www.acfr.usyd.edu.au/homepages/academic/tbailey/software/software.html`

[43] Brown, R.G., Hwang, P.Y.C.: Introduction to Random Signals and Applied Kalman Filtering, 3rd edn. John Wiley and Sons, USA (1997)

[44] Chatterjee, A., Matsuno, F.: A neuro-fuzzy assisted extended Kalman filter-based approach for Simultaneous Localization and Mapping (SLAM) problems. IEEE Transactions on Fuzzy Systems 15(5), 984–997 (2007)

[45] Chatterjee, A.: Differential evolution tuned fuzzy supervisor adapted extended kalman filtering for SLAM problems in mobile robots. Robotica 27(3), 411–423 (2009)

[46] Storn, R.: On the usage of differential evolution for function optimization (1996)

[47] Storn, R., Price, K.: Minimizing the real functions of the ICEC 1996 contest by differential evolution (1996)

[48] `http://www.icsi.berkeley.edu/~storn/code.html` (last accessed June 24, 2008)