Journal of Engineering Research

Volume 7 | Issue 3 Article 13

2023

Enhancing FastSLAM 2.0 performance using a DE Algorithm with Multi-mutation Strategies

Hadeer Adel Attia, Mohamed Arafa, Mohamed TALAAT FAHIM

Journal of Engineering Research (ERJ)
Vol. 7 – No. 3, 2023

ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

Enhancing FastSLAM 2.0 performance using a DE Algorithm with Multi-mutation Strategies

H. A. Attia, Mohamed Saidahmed, M. Arafa

Computers and Control Department, Tanta University, Faculty of Engineering, Tanta, Egypt
Email: Hadir_88122_pg@f-eng.tanta.edu.eg, mohamed_ahmed1@f-eng.tanta.edu.eg, m.Arafa@f-eng.tanta.edu.eg

***Abstract*- FastSLAM 2.0 is considered one of the popular approaches that utilizes a Rao-Blackwellized particle filter for solving simultaneous localization and mapping (SLAM) problems. It is computationally efficient, robust and can be used to handle large and complex environments. However, the conventional FastSLAM 2.0 algorithm is known to degenerate over time in terms of accuracy because of the particle depletion problem that arises in the resampling phase. In this work, we introduce an enhanced variant of the FastSLAM 2.0 algorithm based on an enhanced differential evolution (DE) algorithm with multi-mutation strategies to improve its performance and reduce the effect of the particle depletion problem. The Enhanced DE algorithm is used to optimize the particle weights and conserve diversity among particles. A comparison has been made with other two common algorithms to evaluate the performance of the proposed algorithm in estimating the robot and landmarks positions for a SLAM problem. Results are accomplished in terms of accuracy represented by the positioning errors of robot and landmark positions as well as their root mean square errors. All results show that the proposed algorithm achieves high accuracy than the other compared algorithms in estimating the robot and landmark positions for all the considered cases. It can reduce the effect of the particle depletion problem and improve the performance of the FastSLAM 2.0 algorithm in solving SLAM problem.**

***Keywords*- FastSLAM 2.0, particle filter, Differential Evolution, SLAM Problem.**

I. Introduction

The challenge of a mobile robot navigating an unknown environment is identified as the simultaneous localization and mapping (SLAM) problem. In which, a robot moves from an unknown position in an unknown environment, determining location and building an environmental map at the same time by state estimation and sensor observation [1]. Most robotic applications such as path planning and autonomous manipulation heavily rely on SLAM algorithms. For example, autonomous robots use FastSLAM algorithm to create maps of their surroundings and navigate autonomously through complex environments. There exists also a feature of Google Maps called Google Street View. It uses FastSLAM algorithm to build maps of streets and other public spaces. The constructed maps provide 360-degree panoramic views of any supported location. In addition, most of the location-based services in smartphones mainly rely on FastSLAM algorithm to build maps of indoor environments.

In SLAM classical methods, Kalman filter (KF) and its variants are used as the based methods. They have several issues as follows. The required time to update the extended Kalman filter (EKF) covariance matrices is quadratic in N, where N refers to the number of landmarks. The quadratic complexity of SLAM algorithms has long been recognized as an essential obstacle for scaling them to maps with more than a few hundred features. It also restricts the use of SLAM algorithms to problems with vague landmarks, resulting in a data association problem [2, 3]. In contrast, FastSLAM has a parallelized structure that enables it to achieve the needed performance for big map calculations in real-time applications [4, 5].

There exist several variants of FastSLAM algorithm. The most common ones are FastSLAM 1.0 [4], FastSLAM 2.0 [5], and Unscented FastSLAM (U-FastSLAM) [6]. The FastSLAM 1.0 algorithm estimates the vehicle pose using the generic particle filter (PF) [4], where each particle is coupled with a set of independent extended Kalman filters that are used to determine the position of each feature on the map. In FastSLAM 2.0, some modifications have been made for FastSLAM 1.0 in the selection of proposal distributions and the computation of importance weights. The rest sequence of the algorithm remains identical for both FastSLAM 1.0 and 2.0, including the landmark updates, data association, and resampling operations. FastSLAM 2.0 is considered an enhanced variant of FastSLAM 1.0, as it improves the proposal distribution accuracy, and it uses low-dimensional EKFs in predicting the feature states [5].

In U-FastSLAM, the unscented Kalman filter is applied to update the mean and covariance of the feature state and avoid linearization errors and Jacobean computations in feature estimations. The proposal distribution is determined using the measurement updates of the unscented filter in the particle filter's sampling step. U-FastSLAM can be more accurate in noisy environments. However, unscented Kalman filters are also more complex to implement than regular Kalman filters [6].

FastSLAM algorithms can be applied with a high performance to real-time applications in non-Gaussian environments [4–6]. One of the most crucial advantages of FastSLAM is its accurate estimation of the uncertainty as well as the obtained information about the vehicle's whole route history and its associated map. FastSLAM has an advantage over any other algorithm in solving the SLAM problem of multi-hypothesis data association, which is carried out by using the advantage of the sampling

Doi: 10.21608/ERJENG.2023.235646.1242

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

distribution of each particle [7]. Each particle can contain a different number of feature (or landmark) observations. However, it has sometimes been observed that the accuracy of FastSLAM degrades over time [8]. This degradation occurs when a set of particles, that is used to estimate the robot's pose, loses the diversity among particles. There are two basic reasons for losing particle diversity of FastSLAM [8]. First, when there is a difference between the target distribution and the proposed distribution, it creates improbable particles that incorrectly determine the robot's position. Second, FastSLAM removes improbable particles during the resampling step, leaving only particles with high weights. However, some of the removed particles may contain correct information about the robot position estimation and this information cannot be restored again. This causes a problem known as particle depletion.

In this work, a new approach is proposed to enhance the performance of the FastSLAM 2.0 algorithm and overcome the particle depletion problem. The other sections of the paper are organized as follows. The literature reviews of improving the performance for the FastSLAM algorithm are summarized in Section 2. Its basics are illustrated in Section 3. Then, Section 4 presents the performance enhancing of the FastSLAM 2.0 algorithm using a differential evolution (DE) algorithm with multi-mutation strategies. Section 5 provides the comparisons and simulation results. Finally, Section 6 concludes the work and suggests a future work.

II. Review of Earlier Work

In the field of robotics, and particularly in mobile robot system, SLAM is crucial. Several works have been made to enhance the performance of SLAM algorithms. We will review some of these works considering the FastSLAM algorithm.

In [9], L. Heon-Cheol, P. Shin-Kyu, C. Jeong-Sik, and L. Beom-Hee try to solve the degeneracy by particle cooperation in FastSLAM through using the particle swarm optimization (PSO) to update the robot position after the resampling phase. The results demonstrated that its performance minimized the root mean square error (RMSE) in robot position and map features. In [10], Yi-min Xia and Yi-min Yang used a genetic algorithm (GA) with FastSLAM to solve the sample degradation problems. The improved algorithm achieved higher estimation precision and lower RMSEthan the basic FastSLAM algorithm. In [11], a square root central difference Kalman filter-based FastSLAM (SRCD-FastSLAM) is suggested and improved using a differential evolution (DE) algorithm to handle the particle depletion problem. The results of the study showed that DE-SRCD-FastSLAM is better, in terms of accuracy and robustness, than FastSLAM 2.0, U-FastSLAM, and SRCD-FastSLAM. In [12], GA and PSO are used together to improve the FastSLAM accuracy and overcome the particle depletion problem in. The experiment results show that GA-PSO-FastSLAM efficiently reduced the RMSE occurring during the estimation of the robot and the landmark position. The experiment results show that GA-PSO-FastSLAM efficiently reduced the RMSE occurring during the estimation of the robot and the landmark position. In [13], an ant colony optimization-based resampling approach is proposed to decrease the particle depletion problem. The results show that the enhanced FastSLAM 2.0 based on ant colony optimization can effectively decrease particle scarcity and increase particle distribution. Compared to the previous methods, this enhancement improves the accuracy of SLAM as well as decreases the consumption time. In [14], a novel framework called IFastSLAM is introduced to enhance the performance of FastSLAM based on the PSO algorithm. The authors also use a GA algorithm to increase the diversity of particles in the resampling strategy. Moreover, they improve the conventional PSO algorithm by combining it with the principles of fractional differential and chaotic optimization. The chaotic optimization avoids premature convergence while the fractional differential accelerates algorithm iteration. A global optimization target is proposed for the improved PSO scheme. The experiment results show that the global optimization accuracy is improved, and the robot and landmark estimation errors are reduced. In [15], the robust square-root cubature Kalman filter (RSRCKF) with partial genetic resampling is suggested to improve the performance of FastSLAM. RSRCKF is utilized in the proposed technique to create the FastSLAM proposal distribution and to estimate the environment landmarks. In this method, no prior knowledge of noise statistics is required. Furthermore, it employs a genetic operators-based technique to increase particle diversity. The results indicate that the suggested method gives better accuracy and robust estimation values than the other methods, even with a smaller number of particles and unknown beforehand. The authors in [16] improve the accuracy of FastSLAM in positioning and mapping for an application of a mine robot for fast rescue. The lion swarm optimization approach is used to increase the FastSLAM performance. It uses a division of labor between different individuals to generate the optimized particle set distribution. The particles are distributed in a high probability area, and this helps in solving the particle weight degradation problem. The authors indicate that the diversity of particles is improved as individuals use different foraging techniques in the lion swarm algorithm.

This paper imports evolution mechanisms into the FastSLAM 2.0 algorithm, where an improved differential evolution (DE) algorithm with multi-mutation strategies is used to solve its depletion problem.

III. Basics of FastSLAM 2.0

The SLAM problem is identified as the simultaneous estimation of the vehicle's position and the creation of its observable environment map. The map has N features (landmarks) represented by $\Theta = (\theta_1, \theta_2, \dots, \theta_N)$. The vehicle's path is defined as $x^t = \{x_1, \dots, x_t\}$ where t refers to time index and x_t represents the vehicle's pose at time t.

The basic purpose of SLAM is to recover the best estimate of the robot position x^t over its path and the map landmarks Θ, considering the given collection of noisy

Doi: 10.21608/ERJENG.2023.235646.1242

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

observations z^t, controls $u^t$and set of data associations n^t. This can be explained using the following probabilistic term, which is often known as the SLAM posterior [5]:

$$p(x^t, \Theta \,/\, z^t, u^t, n^t) \quad (1)$$

where $n^t = \{n_1, \dots, n_t\}$ is the data association between features and measurement information, $z^t = \{z_1, \dots, z_t\}$ is the observation sequence, and $u^t = \{u_1, \dots, u_t\}$ is the control input.

To compute the posterior (1), the vehicle provides a probabilistic motion model in the form of the conditional probability distribution. This distribution explains the effects of a control u_t asserted in the time interval $[t-1; t]$ on the resulting pose. The motion model is expressed as follows [5]:

$$p(x_t / x_{t-1}, u_t) \quad (2)$$

where x_t is the present position and x_{t-1} is the vehicle's previous position. The vehicle is also given a probabilistic measurement model that describes how measurements evolve from state. The measurement model is expressed as follows [5]:

$$p(z_t / x_{t-1}, \Theta, n_t) \quad (3)$$

In the FastSLAM 2.0 model, the map features and the vehicle's path can be estimated as if they were separated. A Rao-Blackwellized particle filter is used to realize this estimation [17]. Since the estimation of each landmark is generally independent, FastSLAM 2.0 can be described by the following product of two independent posterior probabilities [5]:

$$p(x^t, \Theta / z^t, u^t, n^t) = p(x^t / z^t, u^t, n^t) \prod_{n=1}^{N} p(\theta_n / x^t, z^t, u^t, n^t) \quad (4)$$

where θ_n is the n-th feature at time t. A particle filter is used by FastSLAM 2.0 to sample the vehicle's path. Each particle has its own map, which consists of N extended Kalman filters. The i-th particle X_t^i includes a path $x^{t,i}$ together with gaussian N landmark estimations, which can be defined by the mean $\mu_{N,t}^i$ and covariance $\Sigma_{N,t}^i$. Each particle can be written in the following form [5]:

$$X_t^i = x^{t,i}, \mu_{1,t}^i, \Sigma_{1,t}^i, \dots, \mu_{N,t}^i, \Sigma_{N,t}^i \quad (5)$$

where $\mu_{1,t}^i$ and $\Sigma_{1,t}^i$ are the mean and covariance of the gaussian distribution, respectively, that characterize the position of the Landmark θ_1. $\mu_{N,t}^i$ and $\Sigma_{N,t}^i$ are the mean and covariance of the gaussian distribution, respectively, that indicate the position of the Landmark θ_N.

The sequence of operations for the FastSLAM 2.0 algorithm can be explained using the following steps:

1. *Sampling new pose:*

Poses are sampled with both motion control u_t and measurement z_t. This is defined by the following sampling distribution, which includes the measurement z_t [5]:

$$X_t^i \sim p(x_t / x^{t-1,i}, u^t, z^t, n^t) \quad (6)$$

2. *Updating the observation landmark:*

The conditional landmark estimates $p(\theta_n / x^t, z^t, u^t, n^t)$ are represented by applying low-dimensional EKFs. The posterior probability of a landmark remains unchanged when it is not observed, unlike when the estimate is updated.

3. *Calculating importance weight:*

Particles taken from the motion model do not match the desired posterior. The importance sampling technique is used to correct this difference. Each sample has a weight that equals the ratio between the target distribution and the proposal distribution. The weighted set of samples is used to generate a new unweighted set of samples [5].

$$w_t^i = \frac{target\ distribution}{proposal\ distribution} = \frac{p(x^{t,i} / z^t, u^t, n^t)}{p(x^{t-1} / z^{t-1}, u^{t-1}, n^{t-1})\ p(x_t^i / x^{t-1,i}, z^t, u^t, n^t)} \quad (7)$$

4. *Resampling:*

Resampling is the final step that selects particles from the temporary particles set. In the resampling process, the temporary particles with large importance weight remain and achieve replication, whereas the particles with small importance weight are rejected or deleted. As a result, all details of the rejected particles, about the robot's path and feature estimations, are lost [9].

IV. Enhancing FastSLAM using a Differential Evolution (DE) Algorithm

Recently, several optimization algorithms have been utilized in particle filters to move particles from the low likelihood region to the high likelihood region. In this paper, an enhanced differential evolution (DE) algorithm is utilized to enhance the performance of the FastSLAM 2.0 algorithm. Multi-mutation strategies are tested during the evolution process of the DE algorithm to accomplish a suitable balance between the exploration and exploitation rates.

A. *The Basic DE Algorithm*

DE is constructed up of four phases: initialization, mutation, crossover, and selection [18-20].

1. *Initialization*

This phase involves randomly selecting a population of NP D-dimension real-valued vectors (NP is the population size and it represents the number of population members) within the optimization problem's search space. Let $X_{i,G}$ represents the ith $(i = 1,2, \dots, NP)$ vector of the population at the current generation G $(G = 0,1, \dots, G_{max})$, and it can be written as [20]:

Doi: 10.21608/ERJENG.2023.235646.1242

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

$$X_{i,G} = [x_{1,i,G}, x_{2,i,G}, \dots, x_{D,i,G}] \quad (8)$$

where $x_{j,i,G}$ is the jth $(j = 1,2,\dots,D)$ component of ith population vector at the current generation G. At $G = 0$, the initial values of the population members are chosen randomly as [20]:

$$x_{j,i,0} = x_j^L + rand_{j,i}.(x_j^U - x_j^L) \quad (9)$$

where $rand_{j,i}$ is a random number that uniformly distributed between [0, 1]. The upper and lower bounds for each parameter are denoted by x_j^U and x_j^L, respectively.

2. *Mutation*

In mutation, the mutant vector $v_{i,G+1}$ is produced for each target vector $x_{i,G}$. It can be produced by one of the common mutation forms listed below [21, 22]:

The first form is characterized by the notation "DE/rand/1",

$$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}) \quad (10)$$

The second one is characterized by the notation "DE/best/1",

$$v_{i,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G}) \quad (11)$$

The third one is characterized by the notation "DE/ rand /2",

$$v_{i,G+1} = x_{1,G} + F.(x_{r2,G} - x_{r3,G}) + F.(x_{r4,G} - x_{r5,G}) \quad (12)$$

The fourth form is characterized by the notation "DE/best/2",

$$v_{I,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G}) + F.(x_{r3,G} - x_{r4,G}) \quad (13)$$

The last form is characterized by the notation "DE/ current-to-best /1",

$$v_{i,G+1} = x_{i,G} + F.(x_{best,G} - x_{i,G}) + F.(x_{r1,G} - x_{r2,G}) \quad (14)$$

where F $\in [0,2]$ is the mutant factor, which is determined by the user, and $x_{best,G}$ is the best individual of the population in the present generation G that has a minimal objective function value.

The indexes i, r1, r2, r3, r4 and r5 are different integers $\in \{1, 2, \dots, NP\}$, where i ≠ r1 ≠ r2 ≠ r3 ≠ r4 ≠ r5. They are randomly generated once for each target vector.

3. *Crossover*

During this stage, the elements of the trail vector $u_{i,G+1}$ is formed by selecting some elements from the target vector $x_{i,G}$ and the remaining elements from the mutant vector $v_{i,G+1}$, according to the following equation [19-22]:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1}, if(rand_j[0,1] \le CR \text{ or } j = j_{rand}) \\ x_{j,i,G}, if(rand_j[0,1] > CR \text{ or } j \ne j_{rand}) \end{cases} \quad (15)$$

where $CR \in [0,1]$ is a control parameter called the crossover rate, j_{rand} is an integer random number $\in [1,2,\dots,D]$, $i = 1,2,\dots,NP$, $j = 1,2,\dots,D$ and $\text{rand}_j \in [0,1]$ is the jth evaluation of a uniform random number.

To make sure that the resulted trial vector lies inside the search domain, the following suggestion is used [23].

$$u_{j,i,G+1} = \begin{cases} x_j^U + \text{rand}_{j,i}.(x_{j,i,G} - x_j^U), \text{if}(u_{j,i,G+1} > x_j^U) \\ x_j^L + \text{rand}_{j,i}.(x_{j,i,G} - x_j^L), \text{if}(u_{j,i,G+1} < x_j^L) \end{cases} \quad (16)$$

4. *Selection*

In this phase, the trial vector $u_{i,G+1}$ and the target vector $x_{i,G}$ are compared, and the vector with the lowest objective function value is selected for the next generation. The selection equation is [20]:

$$x_{i,G+1} = \begin{cases} u_{i,G+1}, if\ \left(f(u_{i,G+1}) \le f(x_{i,G})\right) \\ x_{i,G}\ otherwise \end{cases} \quad (17)$$

where f is the objective function that is desired to be optimized.

B. *DE Algorithm with Multi-mutation Strategies for SLAM Problem*

Several variants of the DE algorithm have been developed with multi-mutation strategies to achieve a good balance between exploration and exploitation rates. We made several trials over the mutation strategies that are commonly used by the different variants of the DE algorithm. We find that the following three mutation strategies are well suited to be used by the FastSLAM 2.0 algorithm for the SLAM Problem.

$$v_{i,G+1} = \begin{cases} x_{best,G} + F_{i,1,G}.(x_{r1,G} - x_{r2,G}) \\ x_{best,G} + F_{i,1,G} \times LDIF_{i,G} \\ mean(x_{r1,G}, x_{best,G}) + F_{i,2,G} \times mean(HDIF_{i,G}, LDIF_{i,G}) \end{cases} \quad (18)$$

The FastSLAM 2.0 algorithm uses these proposed mutations with the DE algorithm to enhance its performance. It tries to optimize particle weights to reduce particle depletion and keep diversity among particles. We refer to the enhanced algorithm as MDE- FastSLAM 2.0. Where $x_{best,G}$ is the best individual in the population. $F_{i,1,G}$ and $F_{i,2,G}$ represent the mutant scaling factors for the suggested mutation strategies. For each mutant vector, the scaling factors are randomly chosen from a predefined range of values. We use $F_{i,1,G} \in F_1$ = [0.1 0.2 0.3 0.4 0.5] and $F_{i,2,G} \in F_2$ = [0.3 0.4 0.5 0.6 0.7], as recommended in [24].

These ranges suit the required search ability for the proposed mutation strategies. $LDIF_{i,G}$ and $HDIF_{i,G}$ represent the two difference vectors with lower and higher objective function values, respectively. These two difference vectors are obtained as follows [24].

Doi: 10.21608/ERJENG.2023.235646.1242

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

$$HDIF_{i,G} = \begin{cases} x_{r2,G} - x_{r3,G}, if\ f(x_{r2,G} - x_{r3,G}) > f(x_{r4,G} - x_{r5,G}) \\ x_{r4,G} - x_{r5,G}, \quad otherwise \end{cases} \quad (19)$$

$$LDIF_{i,G} = \begin{cases} x_{r4,G} - x_{r5,G}, \text{if } f(x_{r2,G} - x_{r3,G}) > f(x_{r4,G} - x_{r5,G}) \\ x_{r2,G} - x_{r3,G}, \quad otherwise \end{cases} \quad (20)$$

where $x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$, $x_{r4,G}$ and $x_{r5,G}$ represent five individuals that are randomly chosen from the present population.

The first mutation strategy is "DE/best/1". The other two mutation strategies were selected from the proposed mutations of the enhanced variant of the DE algorithm that is developed in [24]. These three mutation strategies enhance the search ability of the DE algorithm to suit the SLAM Problem. It is self-evident that the search evolution of the SLAM problem doesn't need for searching with a high exploration rate. Therefore, this has been taken into consideration for the three proposed mutations. The first and second mutation strategies improve the ability of the exploitation search. The third mutation strategy balances the search abilities of exploration and exploitation.

The value of the crossover rate control parameter CR is also affect the performance of the algorithm search ability . It is randomly selected from a predefined range of values to achieve the desired search ability. So, for the case of search with high exploitation search, we use $CR \in CR_1$ = [0.8 0.85 0.9 0.95 1.0], as recommended in [24]. For the case of search with a balanced search of exploration and exploitation, we use $CR \in CR_2$ = [0.4 0.5 0.6 0.7 0.8], as recommended in [24].

All the proposed mutation strategies are evaluated during the first 50 iterations of the computational algorithm. After that, the one that achieves the best results will be exclusively used for the remaining number of iterations.

The fitness function that is used during the optimization process of the enhanced MDE- FastSLAM 2.0 is the same as in [11]:

$$Fitness\ Function = exp\{-\frac{1}{2R_t}\left(Z_{t,act} - \hat{z}^i_{t,pred}\right)^2\} \quad (21)$$

Where R_t is the observation noise covariance, and $Z_{t,act} - \hat{z}^i_{t,pred}$ is the variance between the actual and the predicted observations, respectively. More information is available at [5].

A pseudocode of the enhanced MDE- FastSLAM 2.0 algorithm is shown by Algorithm 1.

Algorithm 1. Pseudo-code of the proposed MDE-FastSLAM 2.0 algorithm.

Start Algorithm
1) Sample new pose of robot for each particle Eq. (6)
2) Update the landmark of the observed features for each particle
3) Compute the new weights of particles using Eq. (7)
4) Optimize the weight of particles using MDE
5) Generate the initial population $P_0 = [x_{1,0}, \dots, x_{NP,0}]$
6) Set the generation number G=0
7) Set the ranges of vectors F_1 , F_2 , CR_1 and CR_2
8) Select randomly the initial values of $F_{i,1,G}$, $F_{i,2,G}$, and CR
9) Evaluate fitness function of all population Eq. (21)
10) While $G < G_{max}$ do
11) For all population do
12) Choose randomly five individuals $x_{r1,G}$, $x_{r2,G}$, $x_{r3,G}$, $x_{r4,G}$, and $x_{r5,G}$
13) Calculate $HDiF_{i,G}$ and $LDiF_{i,G}$ using Eqs. (19-20)
14) Calculate $v_{i,G+1}$ using Eq. (18)
15) End for
16) Calculate $U_{i,G+1}$ using Eq. (15) for all population
17) Calculate $X_{i,G+1}$ using Eq. (17) for all population
18) Next generation (G=G+1)
19) End while
20) Update weight of particles based on optimizer
21) Resampling
End algorithm

V. Experimental Study and Results

To show the performance of the enhanced MDE-FastSLAM 2.0 algorithm, a comparison has been made with other two algorithms that are commonly used to solve the SLAM Problem. It is compared to the standard FastSLAM 2.0 algorithm [5] and the FastSLAM 2.0 algorithm that is enhanced by a differential evolution algorithm (DE-FastSLAM) [11].

This experiment is simulated using MATLAB R2018a Runtime Environment. The FastSLAM 2.0 algorithm is implemented using the MATLAB code developed by Bailey [25].

Fig. 1 shows the tested environment map that is considered a two-dimensional map with 35 landmarks and 17 robot waypoints [10, 11, 14, 16]. The blue trajectory indicates real motion path of the robot, the red 'o' represents the waypoint, the green '*' represents a real landmark, the red '.' represents an estimated landmark, the green triangle represents a real robot, and the red triangle represents an estimated robot.

Doi: 10.21608/ERJENG.2023.235646.1242

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

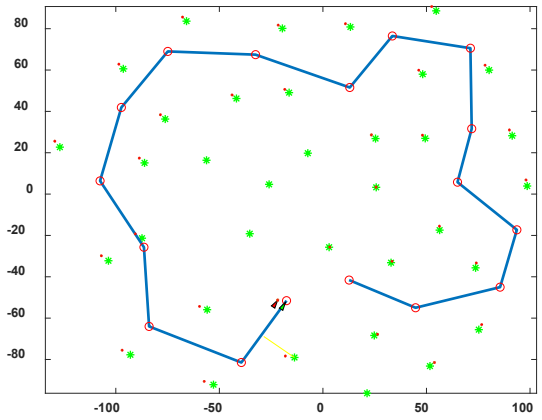


Figure 1. The tested environment map.

The parameter settings during the simulation for the FastSLAM 2.0 algorithm are used as in [10]. These settings are given in Table 1.

Table 1. Parameter settings for FastSLAM 2.0 algorithm

Parameter	Value	Unit
The vehicle speed	1	m/s
The wheelbase	4	M
The control frequency	20	Hz
The maximum steering angle	30 ∗ pi/180	Rad
The maximum rate of change in steer angle	20 ∗ pi/180	rad/s
The speed noise of a vehicle	0.1	m/s
The time interval between observations	0.2	S
The distance of observation	30	M
The distance observation noise	0.1	M
The angle noise of observation	1.0 ∗ pi/180	Rad
Number of independent runs	30	Run

where m, s and rad stand for meter, second and radian, respectively.

The values of the control noise covariance Q and the observation noise covariance R are used as:

$$Q = \begin{bmatrix} 0.1^2 & 0 \\ 0 & \left(\frac{\pi}{180}\right)^2 \end{bmatrix}, R = \begin{bmatrix} 0.1^2 & 0 \\ 0 & \left(\frac{\pi}{180}\right)^2 \end{bmatrix}$$

The same fitness function represented by Eq. (21) is used for the compared algorithms DE- FastSLAM and MDE-FastSLAM 2.0 during the optimization process. Also, we use the same population size (NP) = 10 and the same number of iterations =1000 for these two algorithms. The settings of the remaining parameters of the DE- FastSLAM algorithm are used as in [11], where the mutant factor $F = 0.4$, cross over rate $CR = 0.8$ and the mutation form is "DE/rand/1".

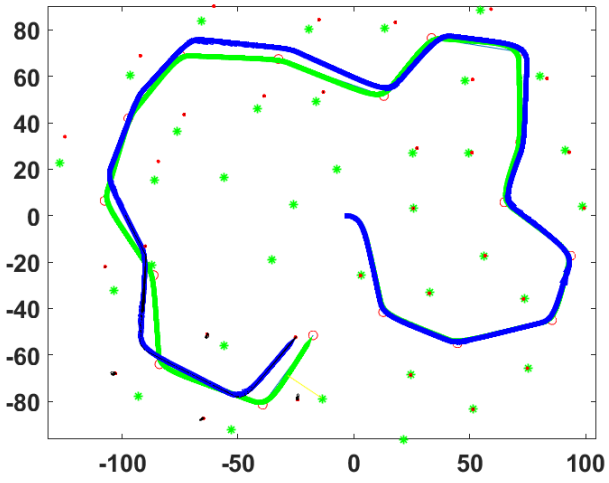
VI. Results

The results in terms of accuracy for the estimation process of paths and landmarks are displayed in this section to evaluate the performance for each one of the compared algorithms.

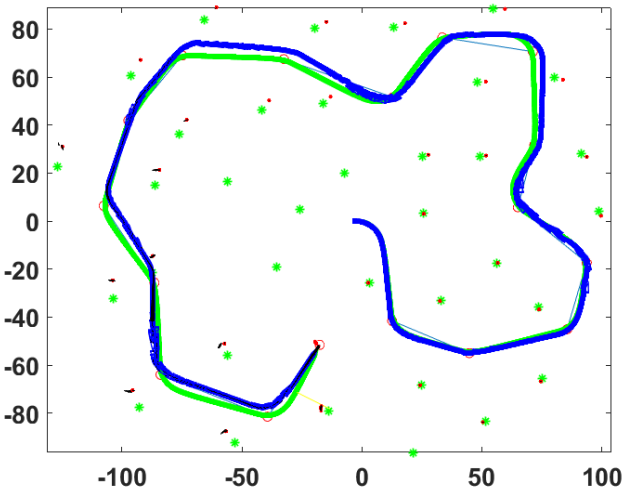
Fig. 2 shows the estimated robot paths and landmark positions when using 10 particles for the three compared algorithms. The green trajectory indicates the actual path and the blue one indicates the estimated path.

Considering the three compared algorithms, we can infer from Fig. 2 that the estimated values by MDE-FastSLAM 2.0 algorithm for landmark positions and paths are mostly consistent with the real values.

In addition to the previous comparison, we evaluate the positioning error (PE) of the robot position for each algorithm using 10 particles [26], as shown in Fig. 3. It provides the absolute error between the actual position and the corresponding average estimations of positions. The PE can be determined using the following equation [14]:

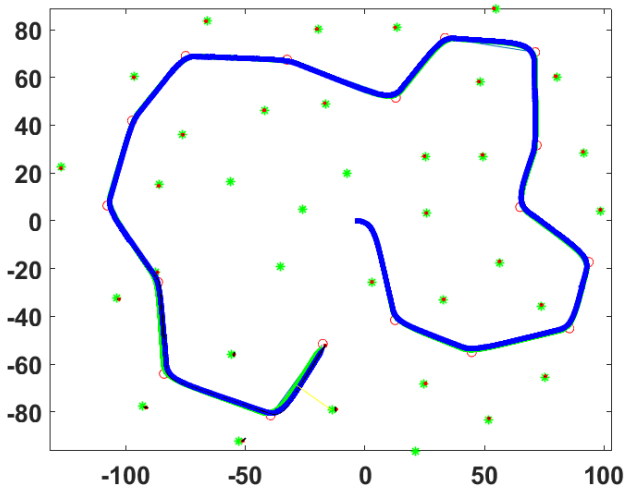


(a) FastSLAM 2.0



(b) DE-FastSLAM

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

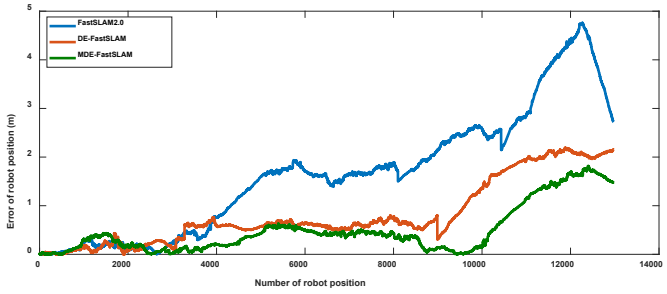


(c) MDE-FastSLAM 2.0

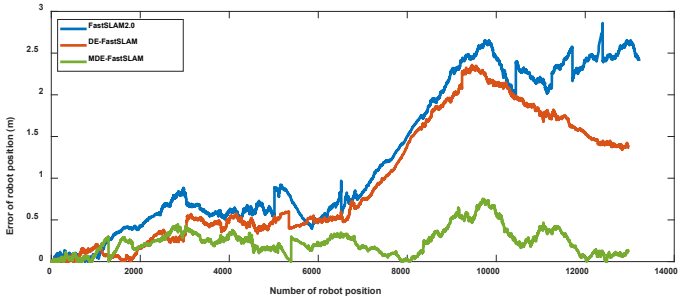
Figure 2. Estimated and real representation for paths and landmarks.

$$PE_t = \sqrt{\left(\sum_{i=1}^{M} x_t^i/M - x_{act}\right)^T \left(\sum_{i=1}^{M} x_t^i/M - x_{act}\right)} \quad (22)$$

where M represents the number of particles, x_{act} is the actual robot pose, and x_t^i is the predicted robot pose for the ith particle.



(a) Robot position error in X-axis



(b) Robot position error in Y-axis

Figure 3. The positioning error of robot poses.

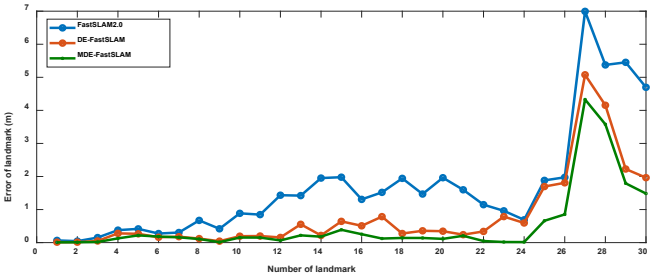
As shown in Fig. 3, MDE-FastSLAM 2.0 algorithm achieves the best estimation accuracy with robot positioning error values less than 1.8 m in X-axis and less than 0.75 m in Y-axis. While the positioning error values for the other compared algorithms FastSLAM 2.0 and DE-FastSLAM are less than 4.8 m and 2.2 m, in X-axis, and less than 2.9 m and 2.4 m, in Y-axis, respectively.

The root mean square error (*RMSE*) of the robot positioning error values along the robot path for the compared algorithms is calculated in Table 2, for 10, 30, and 50 particles. The proposed algorithm has the lowest *RMSE* values of positioning error for all the considered cases of different number of particles.

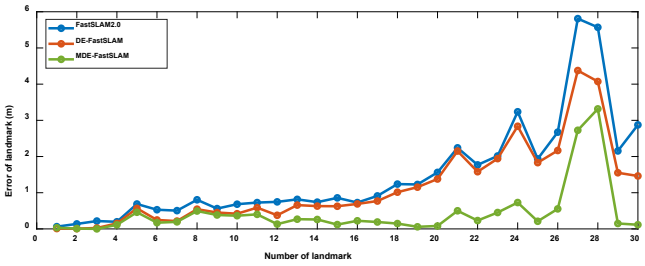
Table 2. RMSE of robot positioning error in meter

Algorithm	Number of Particles		
	10	30	50
FastSLAM 2.0	2.2751	2.1307	1.4711
DE-FastSLAM	1.4988	1.3432	1.1586
MDE-FastSLAM 2.0	0.8653	0.6902	0.6115

Similarly, the positioning error of landmark positions is evaluated for each algorithm using 10 particles, as shown in Fig. 4.



(a) landmark position error in X-axis



(b) landmark position error in Y-axis

Figure 4. The positioning error of landmark positions.

As shown in Fig. 4, MDE-FastSLAM 2.0 algorithm again achieves the highest estimation accuracy with landmark positioning error values less than 4.3 m in X-axis and less than 3.3 m in Y-axis. While the positioning error values for the other compared algorithms FastSLAM 2.0 and DE-FastSLAM are less than 7 m and 5.2 m, in X-axis, and less than 5.8 m and 4.4 m, in Y-axis, respectively.

The *RMSE* of the landmark positioning error values achieved by the compared algorithms for the overall map for is calculated in Table 3, for 10, 30, and 50 particles. The proposed algorithm has the lowest *RMSE* values of positioning error for all the considered cases of different number of particles.

Doi: 10.21608/ERJENG.2023.235646.1242

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

Table 3. RMSE of landmark positioning error in meter

Algorithm	Number of Particles		
	10	30	50
FastSLAM 2.0	1.4785	1.3914	1.2847
DE-FastSLAM	1.2757	1.2115	1.1512
MDE-FastSLAM 2.0	0.5717	0.5412	0.4950

The *RMSE* curves of the robot position for the compared algorithms are shown in Fig. 5. The particle number changes from 0 to 100. It indicates that the proposed algorithm has achieved the minimum *RMSE* values compared to the other two algorithms.

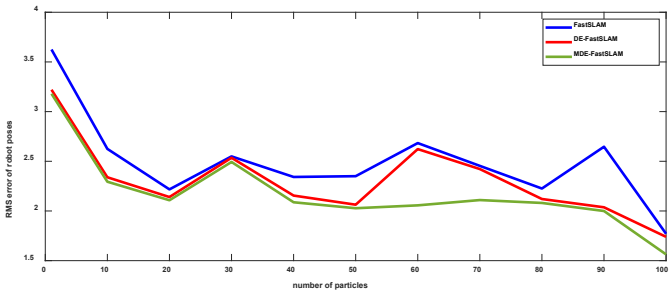


Figure 5. RMSE error of robot poses.

All the previous results demonstrate that the proposed algorithm outperformed the other compared algorithms in terms of accuracy of robot and landmark positions. It was able to reduce the impact of the particle depletion problem while maintaining diversity among particles.

VII. CONCLUSION AND FUTURE WORK

In this work, we proposed an enhanced FastSLAM 2.0 algorithm, denoted by MDE-FastSLAM 2.0, based on an enhanced differential evolution (DE) with multi-mutation strategies to reduce the effect of the particle depletion problem and enhance the performance of FastSLAM 2.0 algorithm in solving the SLAM problem. The enhanced algorithm uses three proposed mutations with the DE algorithm to enhance its performance. It tries to optimize particle weights and keep diversity among particles.

During the optimization process, the proposed mutation strategies are evaluated through the first 50 iterations of the computational algorithm. After that, the one that achieves the best results is used for the remaining iterations. To evaluate the performance of the MDE-FastSLAM 2.0 algorithm, a comparison has been made with other two common algorithms: the standard FastSLAM 2.0 algorithm and the enhanced DE-FastSLAM algorithm. All the compared algorithms are used to estimate the robot and landmarks positions for a SLAM problem. According to the obtained results, the enhanced algorithm outperformed the compared algorithms as it could achieve high accuracy in estimating the robot positions and landmarks through all the considered cases.

For a future work, we can make a hybridization between the enhanced differential evolution using the proposed mutation strategies with a particle swarm optimization (PSO) algorithm. The resulting algorithm can be used with the FastSLAM 2.0 algorithm to further improve the estimation accuracy and increase the particles diversity for SLAM problem.

Funding: The authors should mention if this research has received any type of funding.

Conflicts of Interest: The authors should explicitly declare if there is a conflict of interest.

REFERENCES

[1] L. Wang, Z. Cai, "Progress of CML for Mobile Robots in Unknown Environments", Robot, 2004, 26(4):380-384.
[2] Y. Bar-Shalom and T. E. Fortmann, "Tracking and Data Association", Academic Press, 1988.
[3] J.D. Tard´os, J. Neira, P. Newman, and J. Leonard, "Robust mapping and localization in indoor environments using sonar data", TR TM 2001-04, MIT, 2001.
[4] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem", In: Proceedings of the Eighteenth National Conference on Arti_cial Intelligence; 2002; Menlo Park, CA, USA. Palo Alto, CA, USA: AAAI.pp. 593-598.
[5] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, " FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges", In: Proceedings of IJCAI 2003.
[6] C. Kim, R. Sakthivel, WK. Chung, "Unscented FastSLAM: A robust algorithm for simultaneous localization and mapping problem", In: Proceedings of the IEEE International Conference on Robotics & Automation; 2007; Rome, Italy. New York, NY, USA: IEEE. pp. 2439-2445
[7] M. Montemerlo, S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM", In: IEEE International Conference on Robotics and Automation; 2003; Piscataway, NJ, USA. New York, NY, USA: IEEE. pp. 1985–1991.
[8] T. Bailey, J. Nieto, and E. Nebot, "Consistency of the FastSLAM algorithm", in Proc. IEEE Int. Conf. Robotics and Automation, 2006, pp.424-429.
[9] L. Heon-Cheol, P. Shin-Kyu, C. Jeong-Sik, and L. Beom-Hee, "PSO-FastSLAM: An improved FastSLAM framework using particle swarm optimization", In IEEE International Conference on Systems, Man and Cybernetics, pp. 2763-2768 (2009).
[10] Y.-m. Xia and Y.-m. Yang, "An Improved FastSLAM Algorithm Based on Genetic Algorithms", In Information and Automation. vol. 86, L. Qi, Ed., ed: Springer Berlin Heidelberg, pp. 269-302 (2011).
[11] H. Ankişhan, F. Ari, E. Ö. Tartan, and A. G. Pakfiliz, "Square root central difference-based fast SLAM approach improved by differential evolution", Turkish J. Electr. Eng. Comput. Sci., vol. 24, no. 3, pp. 994–1013, 2016, doi: 10.3906/elk-1307-55.
[12] A. R. Khairuddin, M. S. Talib, H. Haron, and M. Y. C. Abdullah, "GA-PSO-FASTSLAM: A hybrid optimization approach in improving fastSLAM performance", Adv. Intell. Syst. Comput., vol. 557, pp. 57–66, 2017, doi: 10.1007/978-3-319-53480-0_6.
[13] H. Peng, C. Ying, S. Tan, B. Hu, and Z. Sun, "An Improved Feature Selection Algorithm Based on Ant Colony Optimization", IEEE Access, vol. 6, pp. 69203–69209, 2018, doi:
[14] X. Lei, B. Feng, G. Wang, W. Liu, and Y. Yang, "A novel fastSLAM framework based on 2D lidar for autonomous mobile robot", Electron., vol. 9, no. 4, pp. 1–25, 2020, doi: 10.3390/electronics9040695.
[15] R. Havangi, "Robust Square-Root Cubature FastSLAM with Genetic Operators", Robotica, vol. 39, no. 4, pp. 665–685, 2021, doi: 10.1017/S026357472000065X.
[16] D. Zhu, Y. Ma, M. Wang, J. Yang, Y. Yin, and S. Liu, "LSO-FastSLAM: A New Algorithm to Improve the Accuracy of Localization and Mapping for Rescue Robots", Sensors, vol. 22, no. 3, 2022, doi: 10.3390/s22031297.

Journal of Engineering Research (ERJ)
Vol.7 – No. 3, 2023
©Tanta University, Faculty of Engineering
ISSN: 2356-9441 https://erjeng.journals.ekb.eg/ e ISSN: 2735-4873

[17] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks", In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pages 176–183, Stanford, 2000.
[18] K. V. Price, "An Introduction to Differential Evolution", New Ideas in Optimization, McGraw-Hill, London, UK, pp. 79-108, 1999.
[19] M. Ramadas, A. Abraham, "Metaheuristics for Data Clustering and Image Segmentation", Intelligent Systems Reference Library, springer, Switzerland, ISBN: 9783030040963, Vol. 152, 2019.
[20] S. Das and P. N. Suganthan, "Differential Evolution: a Survey of The State-of-the-art", IEEE Transactions on Evolutionary Computation, Vol. 15, No. 1, pp. 4-31, 2011.
[21] K. Opara, J. Arabas, "Comparison of Mutation Strategies in Differential Evolution – A probabilistic Perspective", Swarm and Evolutionary Computation, Vol. 39, pp. 53-69, 2018.
[22] [G. Wu, X. Shen, H. Li, H. Chen, A. Lin, P. N. Suganthan, "Ensemble of Differential Evolution Variants", Information Sciences, Vol. 423, pp. 172–186, 2018.
[23] K. Price, R. Storn, and J. Lampinen, "Differential Evolution: A Practical Approach to Global Optimization ", Berlin, Germany: Springer, 2005.
[24] M. A. Attia, M. Arafa, E. A. Sallam, and M. M. Fahmy, "An Enhanced Differential Evolution Algorithm with Multi-mutation Strategies and Self-adapting Control Parameters", International Journal of Intelligent Systems and Applications, vol. 11, no. 4. pp. 26–38, 2019. doi: 10.5815/ijisa.2019.04.03.
[25] T. Bailey: SLAM Simulation Toolbox. https://www-personal.acfr.usyd.edu.au/tbailey/software/slam_simulations.htm (Last Accessed in 20 September 2023).
[26] X. Chen, H. Sun, and H. Zhang, "A new method of simultaneous localization and mapping for mobile robots using acoustic landmarks", Appl. Sci., vol. 9, no. 7, 2019, doi: 10.3390/app9071352