electronics

MDPI

Article

A Novel FastSLAM Framework Based on 2D Lidar for Autonomous Mobile Robot

Xu Lei [1,2], Bin Feng [1,2], Guiping Wang [1,2,*], Weiyu Liu [1,2] and Yalin Yang [1,2]

[1] The School of Electronics and Control Engineering, Chang'an University, Xi'an 710064, China; xulei@chd.edu.cn (X.L.); fengbin.dev@outlook.com (B.F.); liuweiyu@chd.edu.cn (W.L.); yangyalin1994@outlook.com (Y.Y.)

[2] The National Traffic Information and Control Virtual Simulation Experimental Teaching Center, Chang'an University, Xi'an 710064, China

* Correspondence: gpwang@chd.edu.cn

Received: 24 March 2020; Accepted: 21 April 2020; Published: 24 April 2020

Abstract: The autonomous navigation and environment exploration of mobile robots are carried out on the premise of the ability of environment sensing. Simultaneous localisation and mapping (SLAM) is the key algorithm in perceiving and mapping an environment in real time. FastSLAM has played an increasingly significant role in the SLAM problem. In order to enhance the performance of FastSLAM, a novel framework called IFastSLAM is proposed, based on particle swarm optimisation (PSO). In this framework, an adaptive resampling strategy is proposed that uses the genetic algorithm to increase the diversity of particles, and the principles of fractional differential theory and chaotic optimisation are combined into the algorithm to improve the conventional PSO approach. We observe that the fractional differential approach speeds up the iteration of the algorithm and chaotic optimisation prevents premature convergence. A new idea of a virtual particle is put forward as the global optimisation target for the improved PSO scheme. This approach is more accurate in terms of determining the optimisation target based on the geometric position of the particle, compared to an approach based on the maximum weight value of the particle. The proposed IFastSLAM method is compared with conventional FastSLAM, PSO-FastSLAM, and an adaptive generic FastSLAM algorithm (AGA-FastSLAM). The superiority of IFastSLAM is verified by simulations, experiments with a real-world dataset, and field experiments.

Keywords: FastSLAM; genetic algorithm; resampling; particle swarm optimisation; fractional differential; chaotic; robot

1. Introduction

Owing to the rapid development of autonomous mobile robots, simultaneous localisation and mapping (SLAM) [1] has emerged as a crucial technology in a great many applications, such as self-driving, exploration, and navigation. SLAM can help a robot acquire information on the surrounding environment through a self-positioning process in an unknown environment, and gradually builds an incremental map. The robot performs the SLAM process iteratively in order to eliminate uncertain factors and to estimate the position accurately. Due to the high accuracy and high frequency of lidar, it is very suitable for observation of complex environments. SLAM based on lidar is therefore widely used in practical robot applications. Lidar-based SLAM technologies include 2D lidar SLAM and 3D lidar SLAM. The 3D lidar can identify more details of the environmental space and is suitable for high-precision application scenarios, such as autonomous vehicles, but its application is expensive. The 2D lidar has high measurement accuracy in the two-dimensional plane, which can meet the navigation and positioning requirements of mobile robots. Compared with 3D Lidar, it has lower cost and has the possibility of wide application.

At present, the most widely used SLAM frameworks based on 2D lidar include the approach based on extended Kalman filter (EKF-SLAM) [2,3] and that based on Rao-Blackwellised particle filter (RBPF-SLAM, also known as FastSLAM) [4]. EKF-SLAM ignores the higher-order terms of the Taylor expansion when performing state estimation, resulting in a large uncertainty in the state estimation of the robot. As part of the flow of the algorithm, the covariance matrix must be continuously calculated, which creates increasing computational complexity as the maps expand. In contrast, FastSLAM is a SLAM framework based on a particle filter, which was inspired by early probabilistic experiments by Murphy [5] and Thrun [6]. The particle filter in FastSLAM takes advantage of Monte Carlo sampling technology, which can properly estimate non-linear systems with better state accuracy.

In this paper we focus on the FastSLAM based on 2D lidar. The main drawback of FastSLAM is the particle degeneracy problem, which arises due to insufficient particle diversity during resampling [7]. To overcome the issue of particle degeneracy, researchers have proposed many optimisation algorithms to improve the resampling process in FastSLAM. Cugliari [8] introduced an unscented Kalman filter into FastSLAM and proposed a resampling strategy based on adaptive selection. Liu [9] used an adaptive fading extended Kalman filter to compute the proposal distribution in FastSLAM, and showed that the distribution was closer to the posterior position of the mobile robot and that the degree of particle degradation was reduced. Zhang [10] used a nonlinear adaptive square-root unscented Kalman filter to replace the particle filter in FastSLAM to solve the problem of particle degradation, and found that the accuracy and reliability of the algorithm were improved. While these algorithms have made some progress compared to conventional FastSLAM, they fail to make effective use of the information carried by all particles, and do not use collaborative optimisation methods to solve the issue of particle degeneracy [11].

The genetic algorithm (GA) [12] is a commonly used method of ensuring the diversity of particles before the resampling step. Conventional resampling algorithms suffer from particle degeneracy problems, since higher-weight particles are repeatedly selected and lower-weight particles are deleted after resampling. As a result, the remaining particles cannot closely approximate the true state of the robot [13]. Due to the crossover and mutation steps used, the GA can improve the particle degeneracy problem to a certain extent. Tai-Zhi [14] proposed a new FastSLAM algorithm based on revised genetic resampling and a square-root unscented particle filter, in which a fast Metropolis–Hastings algorithm was used as the mutation operator and was combined with traditional crossover to form a new resampling method. Khairuddin [15] integrated the GA and PSO into FastSLAM, and overcame the particle depletion problem by improving the accuracy of FastSLAM in terms of robot and landmark set position estimation. However, in this approach, the scalar parameters of GA are undetermined, and need to be tuned empirically. To solve these problems, we propose an adaptive genetic resampling algorithm in which the crossover and mutation coefficients are determined adaptively based on the distribution of particle weights. The new resampling algorithm is called AGA-Resampling.

PSO [16] is a group algorithm that optimises a problem by iteratively updating the particle set using local and global optimal values. Global optimal solution can be obtained with high probability. It has been proved that the PSO algorithm can effectively improve positioning accuracy in the problem of positioning of mobile robots [17]. Tdor [18] introduced the PSO algorithm into SLAM for the first time, while Havangi [19] used PSO to optimise the proposal distribution. PSO makes the particles aggregate at locations with high posterior probability before the weights are updated, and the impoverishment of particles can be prevent. Ye [20] proposed a random weight PSO strategy to improve the FastSLAM algorithm, to avoid particle degeneracy and maintain particle diversity. Seung [21] proposed a relational FastSLAM approach that integrated the PSO algorithm into the calculation of the weights and formation of the particles. Zuo [22] proposed a new FastSLAM method based on quantum-behaved PSO to improve the proposal distribution of particles and optimise the estimated particles. Due to the collaborative optimisation of PSO, the accuracy of SLAM was significantly improved. However, there still are some problems in with FastSLAM based on the PSO algorithm: The most typical of these is that the algorithm can easily fall into a local optimum, and another is the low calculation

speed. To further enhance the algorithm, we propose an improved PSO algorithm FCPSO, in which the fractional differential is employed to speed up the evolution of the PSO algorithm and chaotic optimisation is utilised to address the problem of premature convergence.

After particle filtering process in FastSLAM is complete, the map and the robot's path are stored in separate particles, each of which stores different information. The robot autonomously chooses one particle from the particle set to represent the results of FastSLAM. In most cases, the particle that has the highest importance weight is selected to represent the results [23]; however, the particle with the highest weight does not always produce the best result, and this cannot be guaranteed. Nosan [24] has proved that representation by means of particles provides a better result with a higher probability than when the particle with the highest weight is used alone. Hence, we define a virtual particle based on the mean of the particles, and the global optimisation value of the FCPSO algorithm is calculated using the position of this virtual particle. The improved FCPSO algorithm is then introduced into FastSLAM based on AGA-Resampling, which not only maintains the diversity of the particles but also alleviates the particle degeneracy problem in FastSLAM. We call this improved FastSLAM method IFastSLAM.

The remainder of this paper is organised as follows. In Section 2, we describe the standard FastSLAM algorithm, and the proposed improved algorithm is introduced. In Section 3, we discuss simulation experiments and field experiments. Finally, a summary and future prospects are given in Section 4.

2. Methodology

2.1. FastSLAM Algorithm Based on AGA-Resampling

SLAM is the process by which the robot draws the environment map and estimates its position [1]. A schematic diagram of this approach is shown in Figure 1. The mobile robot uses LIDAR to observe unknown landmarks by moving within the environment. The variables described below are defined at time t:

x_t: state vector that represents the location and orientation of the robot
u_t: The control input that drives the robot from state x_{t-1} to state x_t
m_t: A vector describing the position of the ith landmark
z_t: The observation of the ith landmark from the robot

Besides, we define the following collections:

$x_{1:t} = \{x_1, x_2, \ldots, x_t\} = \{x_{1:t-1}, x_t\}$: The trajectory of the robot
$u_{1:t} = \{u_1, u_2, \ldots, u_t\} = \{u_{1:t}, u_t\}$: The control input of the robot
$m = \{m_1, m_2, \ldots, m_n\}$: The collection of landmark
$z_{1:t} = \{z_1, z_2, \ldots, z_t\}$: The collection of landmarks' observation
$n_{1:t} = \{n_1, n_2, \ldots, n_t\}$: The correspondence variable of landmarks

The core idea of the FastSLAM algorithm is to calculate the posterior probability of the robot trajectory and map by factorisation, as shown in

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}, n_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}, n_{1:t}) \prod_{n=1}^{N} p(m_n|x_{1:t}, z_{1:t}, n_{1:t}). \quad (1)$$

In FastSLAM, the Rao-Blackwellised particle filter is used to calculate the robot's trajectory, which is represented by $p(x_{1:t}|z_{1:t}, u_{1:t}, n_{1:t})$. The possible trajectory of the robot is dictated by particles. The map is represented by a series of landmarks that follow a Gaussian distribution, which is updated using the observation model. Based on the premise that the position of the robot is known, the extended Kalman filter is used to estimate the landmarks of the map, which are represented by $p(m_n|x_{1:t}, z_{1:t}, n_{1:t})$.

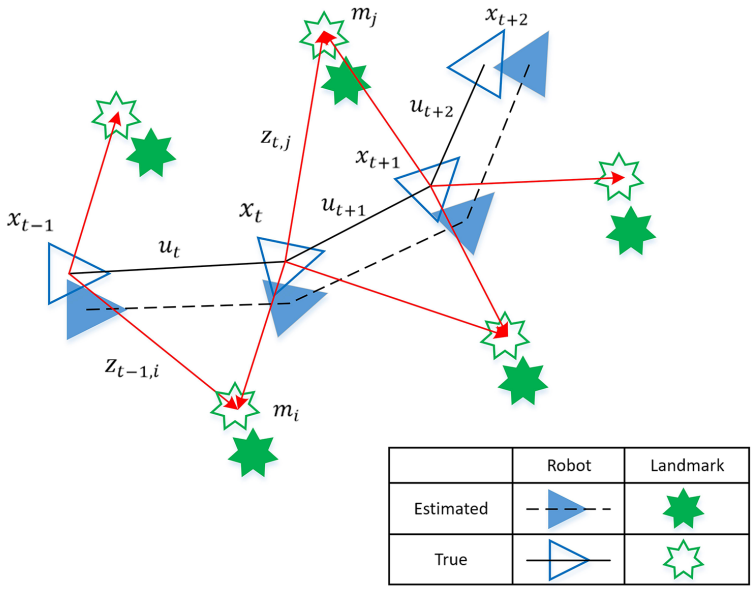


Figure 1. Schematic diagram of simultaneous localisation and mapping (SLAM).

2.1.1. Basic Flow of FastSLAM

Based on information from LIDAR and an odometer, the basic steps of the FastSLAM algorithm are as follows:

1. Initialisation of the particles.

   We initialize N particles, each of which has an initial weight $\omega_0^{(i)} = 0$.
2. Retrieval.

   The initial step involves retrieval of a particle representing the posterior at time $t-1$ and the sampling of the robot's position at time t. The position of the robot is predicted by sampling from a proposal distribution π, and is denoted as $x_t^{(i)}$ for the ith particle. Normally, the proposal distribution generally adopts the probabilistic motion model $p(x_t|x_{t-1}, u_t)$.
3. Importance weight.

   The importance weight for the ith particle $\omega_t^{(i)}$ represents the proportion between the target distribution $p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1}, n_{1:t})$ and the proposal distribution π mentioned above. It is defined as follows:

$$\omega_t^{(i)} = \frac{p(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1}, n_{1:t})}{\pi(x_{1:t}^{(i)}|z_{1:t}, u_{1:t-1}, n_{1:t})}, i = 1, \ldots, N \tag{2}$$

4. Resampling.

   We define the effective number of particles N_{eff} as follows:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(\omega_t^{(i)})^2}. \tag{3}$$

   By comparing N_{eff} with the threshold set in advance, it can be determined whether to perform resampling. The threshold selected here is $N_{th} = N/2$, where N denotes the total number of particles. When $N_{eff} > N_{th}$, resampling is performed, in which higher weighted particles are copied and lower weighted particles are discarded.
5. Measurement update.

The posterior of the map feature estimation $p(m_n|x_{1:t}, z_{1:t})$ is updated based on the robot trajectory $x_{1:t}$ and the observation $z_{1:t}$ of each particle.

The process of the standard FastSLAM is graphically depicted in Figure 2.

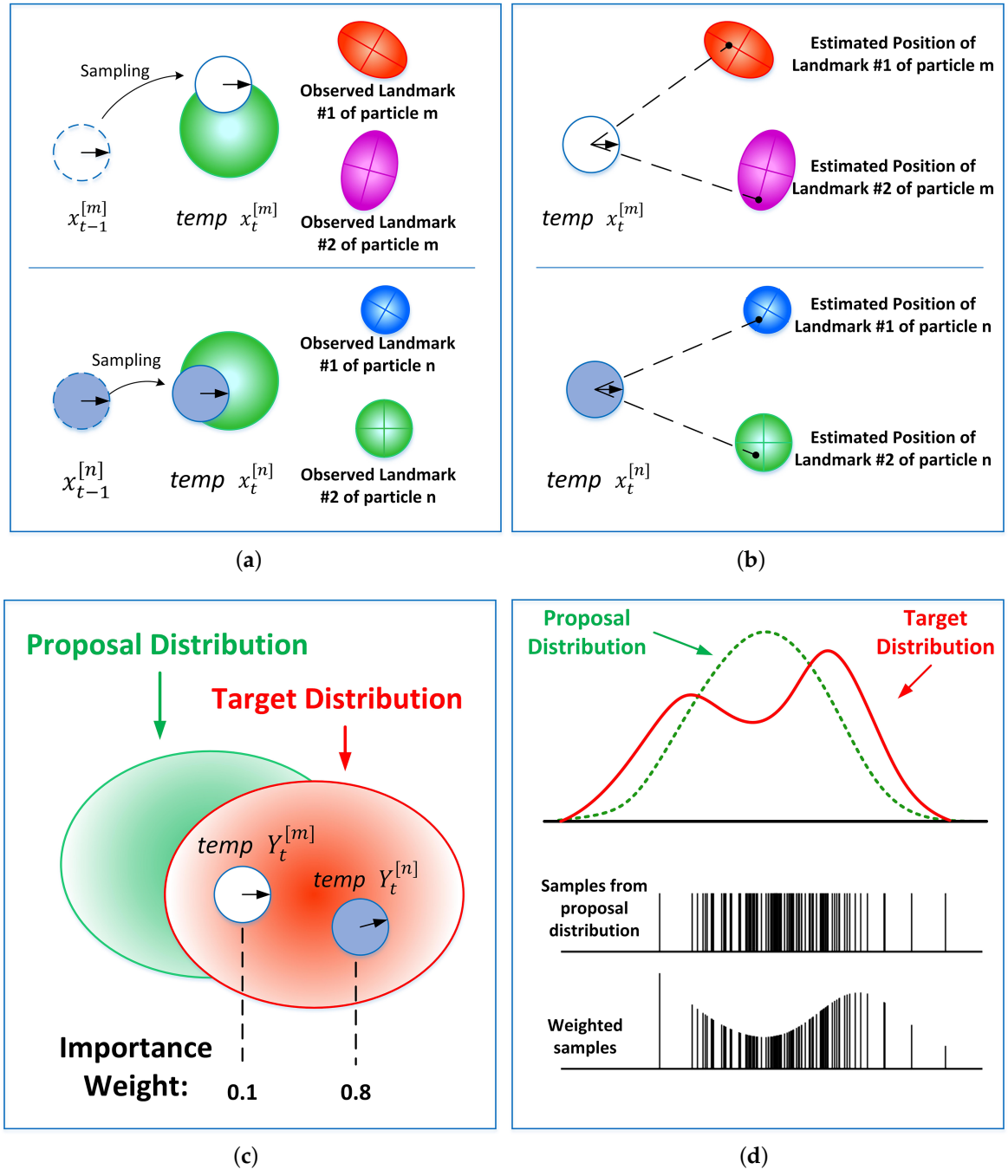


Figure 2. The process of the standard FastSLAM. (**a**) Sampling. (**b**) Measurement. (**c**) Importance weight. (**d**) Resampling.

2.1.2. Particle Degeneracy Problem in FastSLAM

In the resampling phase, it is essential to remove those particles with large estimation errors. However, the resampling process leads to insufficient particle diversity: Only the higher weighted particles are replicated, while the lower weighted particles are rejected, and the diversity of the particles is therefore reduced.

The particle degeneracy problem can be effectively solved by adding those particles that can accurately estimate the robot's position. However, since the position of the robot is unknown, it is difficult to maintain good particles. Only when the particle swarm is able to share the most likely position of the robot can a sufficient number of good particles be maintained.

2.1.3. Adaptive GA Resampling

In this section, we propose an adaptive strategy by introducing an improved resampling method based on the GA [25]. Before resampling is applied, higher weighted particles are used to correct for lower weighted particles through the crossover and mutation operations of the GA. The adaptive strategy is used to determine the crossover degree and mutation probability for the GA. Compared with traditional particle filtering, this resampling method based on an adaptive GA can effectively improve the diversity of the particle swarm, and this can to some extent solve the particle degradation problem of FastSLAM.

We assume that the particle sets are represented as $x_t^{(i)}, \omega_t^{(i)}$ (i = 1, ..., N), where $\omega_t^{(i)}$ represents the normalised particle weights. We sort particle weights $\omega_t^{(i)}$ (i = 1, ..., N) in descending order and store them in W:

$$W = \left\{ \omega_t^{(1)}, \omega_t^{(2)}, \ldots, \omega_t^{(N)} \right\}. \tag{4}$$

Then we define a threshold W_T based on N_{eff} :

$$W_T = W(N_{eff}). \tag{5}$$

where $W(i)$ represents the ith weight in the set W.

The particles are divided into two groups by W_T as follows:

$$x_t^{(i)} \in \begin{array}{ll} X_L, & if \quad \omega_t^{(i)} \leq W_T \\ X_H, & if \quad \omega_t^{(i)} > W_T \end{array} \tag{6}$$

where X_L and X_H are particle sets that contain the lower and higher weight particles, respectively. We assume that $x_{t,L}^{(i)}$ and $x_{t,H}^{(i)}$ represent the particles from X_L and X_H, respectively. If $x_{t,c}^{(i)}$ represents the modified particles, the formulation of the crossover operator can be given by:

$$x_{t,c}^{(k)} = \mu x_{t,L}^{(i)} + (1-\mu) x_{t,H}^{(j)} \tag{7}$$

where $i = 1, \ldots N_L$, $j = 1, \ldots N_H$, and $k = i + j$. N_L and N_H denote the numbers of particles contained in X_L and X_H, respectively. The parameter $\mu \in [0, 1]$ represents the crossover degree of the particles. The greater the value of μ, the more information will be transferred from $x_{t,L}^{(i)}$ to $x_{t,c}^{(i)}$ and the less information will be transferred from $x_{t,H}^{(j)}$ to $x_{t,c}^{(i)}$. Especially when μ=1, $x_{t,c}^{(i)} = x_{t,L}^{(i)}$, no crossover occurs.

A mutation operator is applied to further increase the diversity of the particles. We assume that $x_{t,m}^{(i)}$ represents the mutated particles. It may happen on the modified particles $x_{t,c}^{(i)}$ according to:

$$x_{t,m}^{(i)} = \begin{cases} 2x_{t,H}^{(j)} - x_{t,c}^{(i)}, & if \quad r_l \leq P_M \\ x_{t,c}^{(i)}, & if \quad r_l > P_M \end{cases} \tag{8}$$

where r_l is a random variable drawn from a uniform distribution on [0,1]. P_M denotes the mutation probability, which needs to be set in advance; if P_M = 0, no mutation will occur.

An adaptive selection strategy is proposed here in order to determine the crossover degree μ and the variation probability P_M. It is well known that the smaller the variance in the particle weights, the better the approximation to the true posterior distribution. We assume that σ^2 represents the variance in the particle weights. Mathematical theorems have shown that for a sequence in the range

[0,1], the variance is between zero and 1/4 [26], and the proof of this theorem is listed in Appendix A. We can therefore draw the conclusion that when the weights of the particles are within the range [0,1], the variance σ^2 is within the range [0,1/4]. The higher the value of σ^2 , the more crossover needs to be performed. When $\sigma^2 = 0$, the quality of the particles is good enough that no crossover is necessary. The relationship between μ and σ^2 can therefore be represented as:

$$\mu = 1 - 4\sigma^2 \tag{9}$$

When $\sigma^2 = 0$, the parameter $\mu = 1$, meaning that no crossover occurs. Conversely, when $\sigma^2 = 1/4$, the parameter $\mu = 0$, meaning that a higher degree of crossover is needed to mitigate the degeneracy problem of the particles.

Mutation is applied to the particles $x_{t,c}^{(i)}$ which are obtained from crossover with a certain probability P_M. In general, $P_min = 0.005$ and $P_max = 0.01$ [27].

When crossover is applied, the weights of the particle set also change. The variance in the new particle weights is expressed as σ_*^2. Similarly, the probability of mutation can also be determined based on σ_*^2; the greater the value of σ_*^2, the more mutation need to be performed, and the higher the probability of mutation P_M. In particular, when $\sigma_*^2 = 1/4$, the mutation probability $P_M = P_{max}$, and when $\sigma_*^2 = 0$, the mutation probability $P_M = P_{min}$. We therefore propose an adaptive selection strategy for P_M as follows:

$$P_M = P_{min} + (P_{max} - P_{min}) \cdot 4\sigma_*^2 \tag{10}$$

In summary, we propose an improved resampling method based on the adaptive GA, which is referred to here as AGA-Resampling. This approach is used to replace the classical resampling of FastSLAM.

2.2. FastSLAM Optimised by FCPSO

2.2.1. PSO Improved by Fractional Differential

The PSO algorithm is derived from animal predation behavior. In this algorithm, the particle swarm is randomly initialised using the fitness function to evaluate the solution, and the optimal solution is found by iteration [28].

We assume that $X_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$ represents the position of the ith particle, $P_i = (p_{i1}, p_{i2}, \ldots, p_{iN})$ represents the local optimal of the particles, and $P_g = (p_{g1}, p_{g2}, \ldots, p_{gN})$ represents the global optimisation of the particles. After t iterations, the velocities of the particles are updated by Equation (11) and the positions are updated using Equation (12):

$$v_{id}^t = \omega_i v_{id}^{t-1} + c_1 r_1 (p_{id} - x_{id}^t) + c_2 r_2 (p_{gd} - x_{id}^t) \tag{11}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^t. \tag{12}$$

In Equation (11), d denotes the dimension, where $d = 1, ..., N$; v_{id} is the d-dimensional component of the velocity of the ith particle; x_{id} is the d-dimensional component of the position of the ith particle; p_{id} is the d-dimensional component of P_i; p_{gd} is the d-dimensional component of P_g; t is the number of iterations; ω is the inertia factor that determines the global and local optimisation ability; c_1 represents the individual learning factor of each particle; c_2 represents the collective learning factor for each particle; and r_1 and r_2 are random numbers between zero and one. In Equation (12), because the sample period between t and $(t + 1)$ is 1, the moving distance in unit time is $v_{id}^t \times 1$, so the position x_{id}^{t+1} can be calculated with the sum of a position x_{id}^t plus a velocity v_{id}^t. The standard PSO algorithm has the problems of slow convergence and easy to fall into local convergence. So it is necessary to make improvements.

The convergence speed of PSO is limited by its algorithm structure. Since fractional-order model has superior memory characteristics, fractional differential theory is of great assistance to the dynamic

evolution of particles in a particle swarm [29]. In this part, we introduce the fractional differential into the PSO algorithm to change the convergence speed.

The fractional differential equation [30] for the discrete-time domain defined by Grunwald–Letnikov is as follows:

$$D^{\alpha}[x(t)] = \frac{1}{T^{\alpha}} \cdot \sum_{k=0}^{r} \frac{(-1)^{k}\Gamma(\alpha+1)x(t-kT)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \tag{13}$$

where Γ denotes the gamma function, T denotes the sampling period, and r denotes the cut-off order of the equation.

The speed update formula for the PSO algorithm is rearranged as follows:

$$v_{id}^{t} - \omega_i v_{id}^{t-1} = c_1 r_1 (p_{id} - x_{id}^{t}) + c_2 r_2 (p_{gd} - x_{id}^{t}). \tag{14}$$

Assuming $\omega = 1$, this equation is a discrete form of differentiation. Assuming $T = 1$ and following Pires et al. [30] , we transform Equation (14) as follows:

$$D^{\alpha}[x(t)] = c_1 r_1 (p_{id} - x_{id}^{t}) + c_2 r_2 (p_{gd} - x_{id}^{t}). \tag{15}$$

Non-integer calculus cannot be directly processed by existing simulation tools, so it is necessary to use finite-dimensional ideas to approximate fractional calculus [31]. Only the first four items are considered, and Equation (15) can be expressed as:

$$v^{t} = \alpha v^{(t-1)} + 1/2\alpha v^{t-2} + 1/6\alpha(1-\alpha)v^{t-3} + 1/24\alpha v(1-\alpha)(2-\alpha)^{t-4} + c_1 r_1 (p_{id} - x_{id}^{t}) + c_2 r_2 (p_{gd} - x_{id}^{t}). \tag{16}$$

By applying the idea of the fractional differential, the order of velocity differential α can be generalised to a fraction between zero and one, which will result in a more stable change and a longer memory effect. The value of α greatly affects the specific performance of the PSO algorithm: The dynamic performance of the PSO algorithm is weakened when α is small, while if the value of α is relatively large, the particle diversity will be increased and the global optimal value will be obtained easier. Based on previous experiments [32], the fractional order α is set to 0.6 where the algorithm converges at the fastest rate.

2.2.2. PSO Improved by Chaotic Optimisation

PSO is a relatively simple and fast-running algorithm. However, it cannot search for the global optimal solution when all particles move toward a local optimum, and this gives rise to the premature convergence phenomenon. The introduction of chaotic optimisation into the search algorithm can help particles escape from local optima by generating many neighboring solutions for the local optimum [33]. Hence, we use a chaotic search to update a certain particle whenever it is judged that PSO is showing premature convergence.

The judgment of premature convergence is based on the variance in the fitness of the particle swarm [34]. The variance in the fitness of the particle swarm is denoted by σ_f^2:

$$\sigma_f^2 = \sum_{i=1}^{N} \frac{f_i - f_{avg}}{f}, \tag{17}$$

$$f = \begin{cases} max_{1 \le i \le m} \mid f_i - f_{avg} \mid, & max_{1 \le i \le m} \mid f_i - f_{avg} \mid > 1 \\ 1, & others \end{cases} \tag{18}$$

where f is the normalization factor, f_i the fitness of the ith particle, and f_{avg} the average fitness of the particle swarm.

The fitness variance σ_f^2 reveals the aggregation degree of the particle swarm. The smaller the value of σ_f^2, the more aggregated the particle swarm. When σ_f^2 is less than a constant C, chaotic operation is performed.

The chaotic search space in this work is determined by the optimal position $p_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$, which is found when the PSO algorithm falls into local convergence. The chaotic variables is generated by the Logistic function [35] as show in Equation (19).

$$y_{q'} = 4y_q(1 - y_q), q = 1, 2, \ldots, n. \tag{19}$$

N chaotic variables $y_{q'}(0 < y_{q'} < 1)$ can be obtained by substituting n initial values $y_q(0 < y_q < 1)$ with small differences into Equation (19). A solution vector to the objective function is then generated according to Equation (20).

$$z_{q'} = y_{q'} - p_{gq}, q = 1, 2, \ldots, n. \tag{20}$$

In summary, when the particles fall into a local optimum, the following steps are performed: (1) Use the Logistic function to generate the chaotic sequence with initial chaotic variable with values in the range [0,1]. (2) Calculate the fitness of each chaotic sequence and record the sequence with the best fitness value. (3) Replace the local optimal particle of the current particles with the chaotic sequence with the best fitness.

The standard PSO is improved by both fractional differential theory and chaotic optimisation. This improved algorithm is referred to here as fractional differential and chaotic particle swarm optimisation (FCPSO). The particle update speed is accelerated, and the hereditary nature of PSO is increased. The particle swarm is also updated when premature convergence occurs, which increases the diversity of the particle swarm.

2.2.3. The Proposal of the Virtual Particle

In general particle filter algorithm based on a standard PSO algorithm, the particle fitness in the PSO algorithm is calculated based on the weight of the particle, and the particle with the largest weight is taken as the global optimisation target [36]. However, it is proved that the mean value of the particle positions is closer to the real location than the particle with the maximum weight [24]. We therefore define a virtual particle according the mean value of the particle positions in this part. The virtual particle is not a real particle in PSO, and it is just used to determine a global optimum for PSO optimisation.

The position of the virtual particle is calculated as follows:

$$x_{vir} = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{21}$$

where x_i represents the initial position of the ith particle.

We introduce FCPSO optimisation into the FastSLAM algorithm. In particular,the fitness function is calculated based on the position of the particle instead of the importance weight:

$$f_i = \| x_i - x_{vir} \|, \tag{22}$$

For each particle, we calculate the local optimum as follows:

$$P_i = x_i. \tag{23}$$

We select the virtual particle as the global optimum target rather than the largest weighted particle as follows:

$$P_g = x_{vir}. \tag{24}$$

The main process of FCPSO is shown in Figure 3.

We use the FCPSO algorithm to optimise the FastSLAM algorithm. First, we use the particle filter based on AGA-Resampling to resample the possible robot position. We then define the virtual particle and use it as the optimisation target of FCPSO, which attracts particles to move to the actual position of the robot. The proposed method, which uses FastSLAM based on AGA-Resampling and is further optimised by FCPSO, is called IFastSLAM. This approach is different from the standard FastSLAM scheme because it uses an additional process to compensate for particle degeneracy. This compensation process serves as a key strategy. The overall scheme of the algorithm is shown in Figure 4, and pseudo-code is given in Algotithm 1.

IFastSLAM has several advantages: AGA-Resampling can effectively improve the diversity of particles, and the iteration speed of PSO with fractional differential optimisation is increased, further improving the speed of SLAM. In addition, PSO optimised by chaos can avoid local convergence and can effectively improve the accuracy of SLAM.

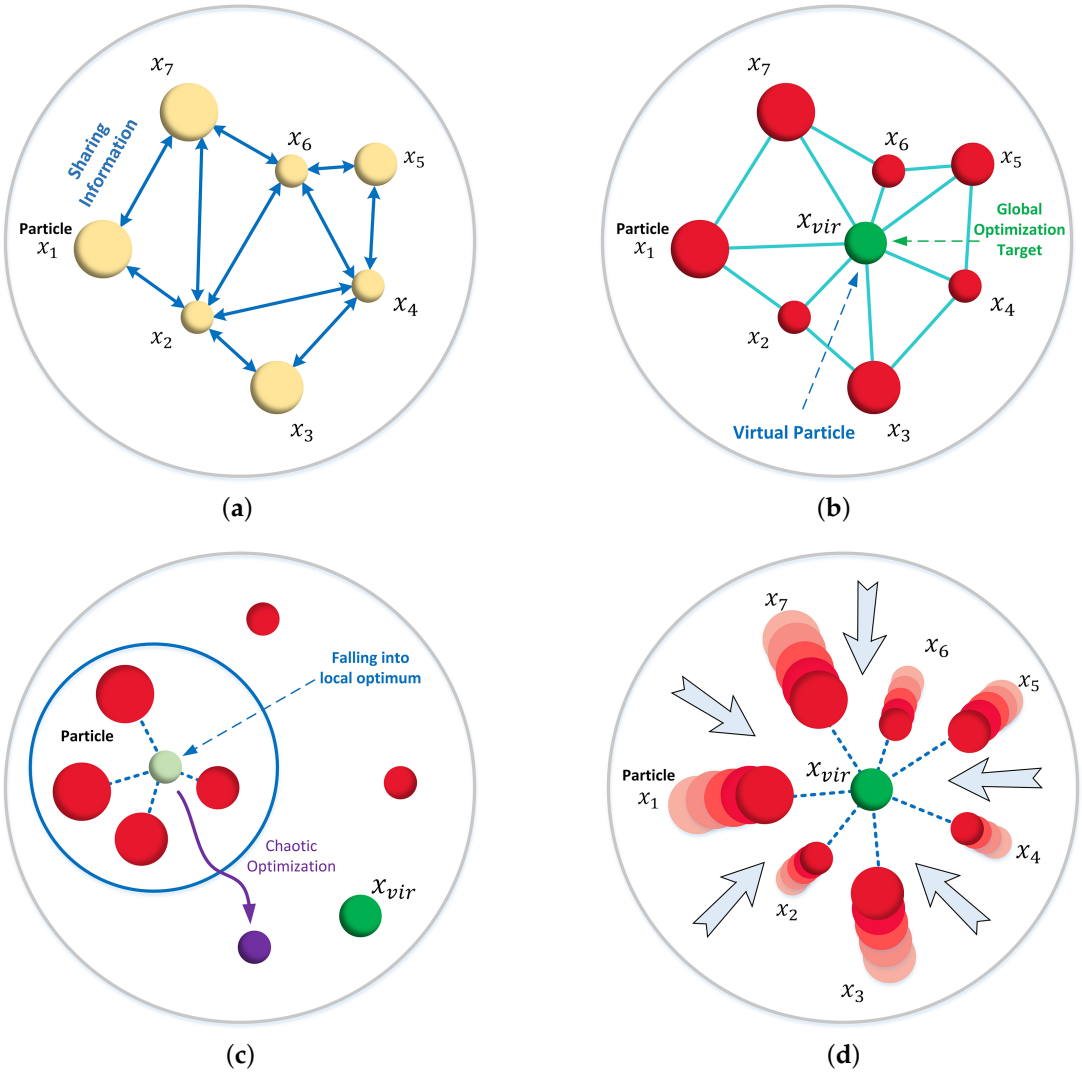


Figure 3. Particle update process of FCPSO. (**a**) Information sharing between particles. (**b**) Define the virtual particle to get the global optimisation goal. (**c**) Deal with particle premature convergence using chaotic optimisation. (**d**) Particles move toward global optimisation target.

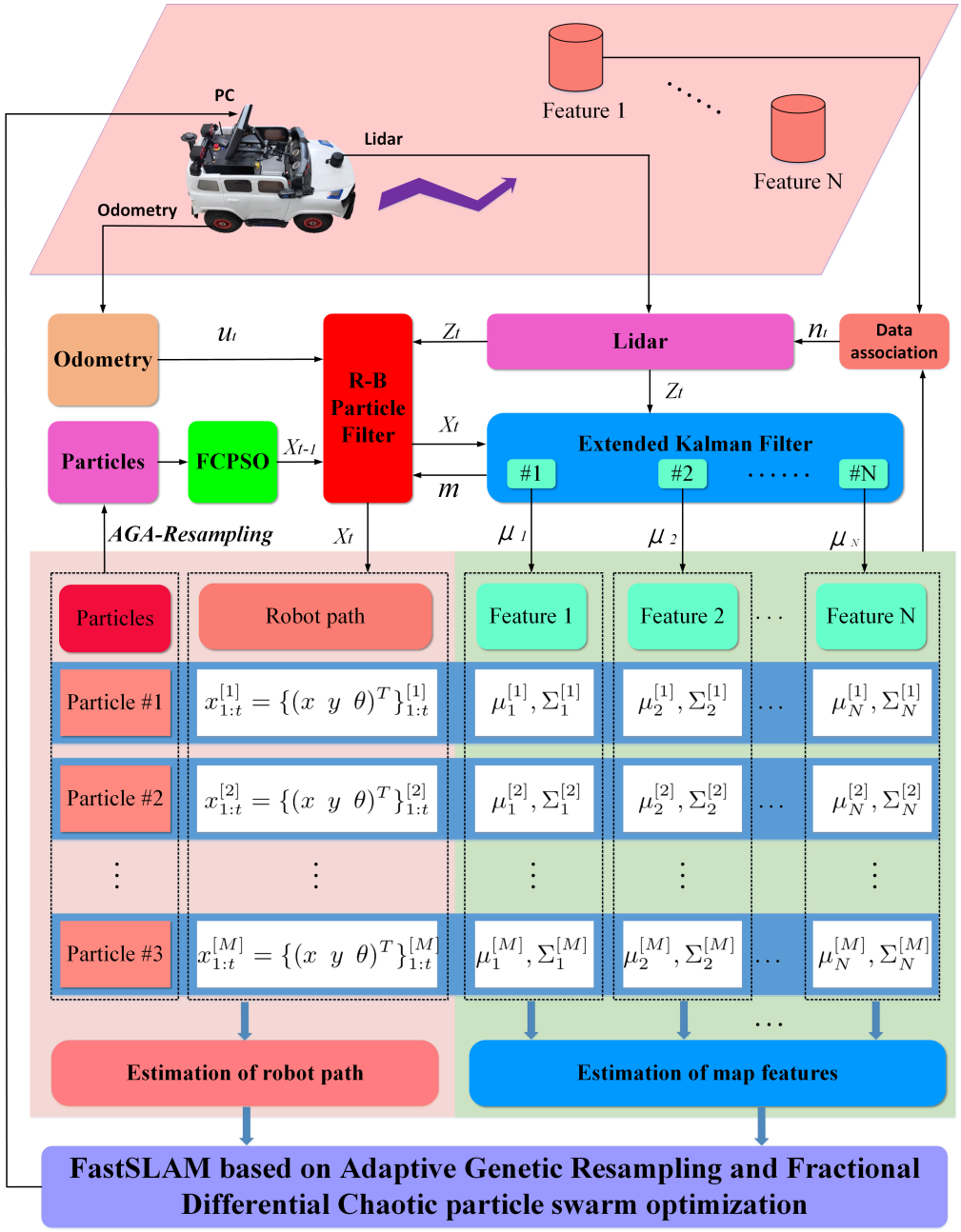


Figure 4. Scheme of IFastSLAM.

Algorithm 1: IFastSLAM

for $m = 1$ *to* N **do**
  Retrieval. Retrieval m_{th} particle from the particle set;
  Prediction. Sample a new pose $x_t^{[k]} \sim p(x_t|x_{t-1}, \mu_t, n_{1:t})$;
  Measurement update. For each observed feature z_t^i identify the correspondence j for the measurement z_t^i, and incorporate the measurement z_t^i into the corresponding EKF, by updating the mean $\mu_{j,t}^{[k]}$ and covariance $\sum_{j,t}^{[k]}$;
  Importance weight. Calculate the importance weight $\omega_t^{(i)}$ for the new particle;
  AGA-Resampling. Use crossover and mutation operators [Equations (7) and (8)] where the coefficients and mutation probability are given by [Equations (9) and (10)] to modify the robot state;
end
Define the virtual particle: calculate the position of the virtual particle. [Equation (21)] ;
for $m = 1$ *to* N **do**
  Initialize the velocity of the m_{th} particle;
  Initialize the velocity of the m_{th} particle;
  Initialize the fitness value of the m_{th} particle;
end
for $k = 1$ *to* N **do**
  for $m = 1$ *to* N **do**
    Calculate the fitness value of the m_{th} particle. [Equation (22)] ;
  end
  Determine the global optimal. [Equation (24)] ;
  for $m = 1$ *to* N **do**
    Determine the local optimal of the m_{th} particle. [Equation (23)] ;
    Update the velocity of the m_th particle using Fractional Differential. [Equation (16)] ;
    Update the position of the m_th particle. [Equation (12)] ;
  end
  Calculate the aggregation extent of the particle set. [Equation (17)] ;
  if *premature convergence occurred* **then**
    Implement Chaotic optimisation. [Equations (19) and (20)];
  end
end

3. Experiments

In this section, we perform experiments with the simulation environment, the Victoria Park dataset, and field experiments, in order to study the performance of the improved algorithm. We performed simulation experiments using Matlab on Intel(R) Core (TM) i5-8250 CPU@1.6 GHz RAM 8 GB PC. We used a robot equipped with LiDAR (SLAMTEC RPLIDAR) and an odometer for field experiments.

3.1. Simulation

3.1.1. Simulation Setup

The simulation was performed using MATLAB to verify the accuracy and stability of IFastSLAM. We developed the SLAM simulator based on the source code published by Bailey [37].

In this work, we assumed that the robot moved at a constant speed. The model for the motion is as follows:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} + \begin{bmatrix} v\Delta t cos(G+\theta_t) \\ v\Delta t sin(G+\theta_t) \\ v\Delta t cos(G)/WB \end{bmatrix} + W_t \tag{25}$$

where $[x_t, y_t, \theta_t]^T$ is the vector of the robot pose; v is the robot speed; G is the steering angle of the robot; WB is the wheelbase of the robot; Δt is the sampling time interval; W_t is the system noise, and $W_t \sim N(0, Q)$.

The observation model used in the simulation experiment is as follows:

$$z_t = \begin{bmatrix} \rho_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \sqrt{(m_x - x_t)^2 + (m_y - y_t)^2} \\ atan(m_y - y_t, m_x - x_t) - \theta_t \end{bmatrix} + V_t \tag{26}$$

where $[m_x, m_y]^T$ is the coordinate of the ith landmark detected by the robot; ρ_i and β_i represent the distance and angle, respectively, of the ith landmark from the current robot position; V_t is the observation noise; and $V_t \sim N(0, R)$.

The simulation environment is illustrated in Figure 5a. It is in effect a search area of size 250×200 m. In this simulation experiment, 17 waypoints and 35 landmarks were randomly set. The red trajectory represents the designated simulation path, the red 'o' is the waypoint, and the blue asterisk is the stationary landmark. The robot starts from (0,0) and moves counterclockwise along the red trajectory shown in Figure 5a to obtain the estimated trajectory and the estimated landmark. Figure 5b shows a screenshot of the process of simulation at a certain moment. In Figure 5b, the black triangle represents the robot, and the green circle represents the observation range of the robot, which is centered on the robot. The robot can observe landmarks within a certain measuring angle inside the green circle. The estimated trajectory is represented by a black line, and the estimated landmark is denoted by a red asterisk.

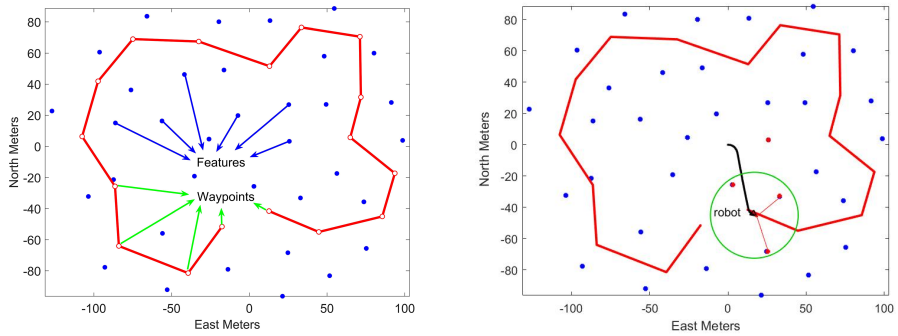


Figure 5. Simulation interface. (**a**) Simulation environment. (**b**) A screenshot of the process of simulation.

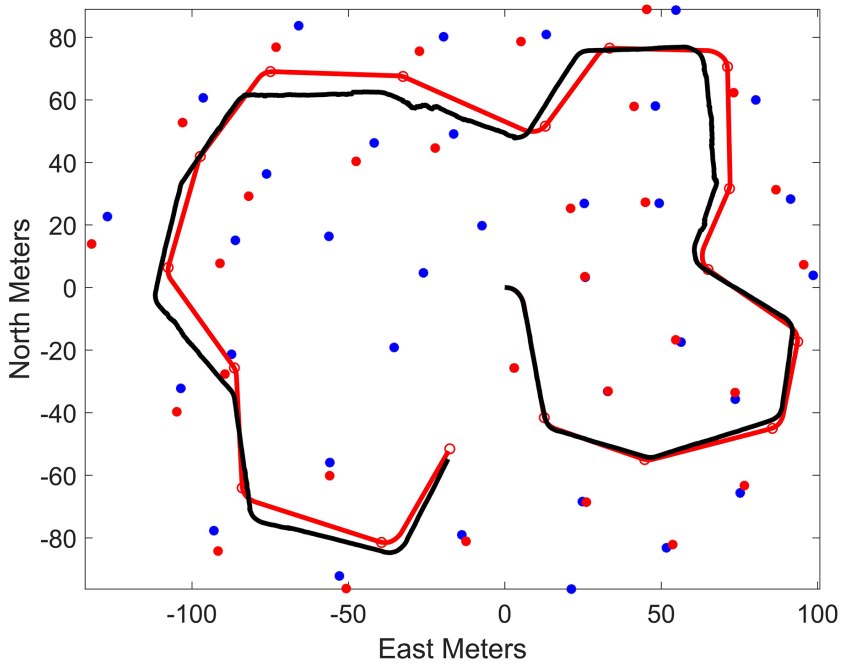
The relevant parameters in the simulation are as follows. The initial position of the robot is $[0, 0, 0]^T$; its speed is $v = 3$ m/s; the maximum steering angle is 20°; The sampling time is 0.025 s; error in the speed is $\sigma_v = 0.3$ m/s; the sampling time of the laser radar is 0.2 s; the maximum measuring distance is 30 m, the measurement error in the distance is $\sigma_\rho = 0.1$ m; and the error in the angle is $\sigma_\beta = 1°$; The systematic noise Q and the observation noise R are as follows:

$$Q = \begin{bmatrix} 0.3^2 & 0 \\ 0 & (\frac{3\pi}{180})^2 \end{bmatrix}, R = \begin{bmatrix} 0.1^2 & 0 \\ 0 & (\frac{\pi}{180})^2 \end{bmatrix} \tag{27}$$

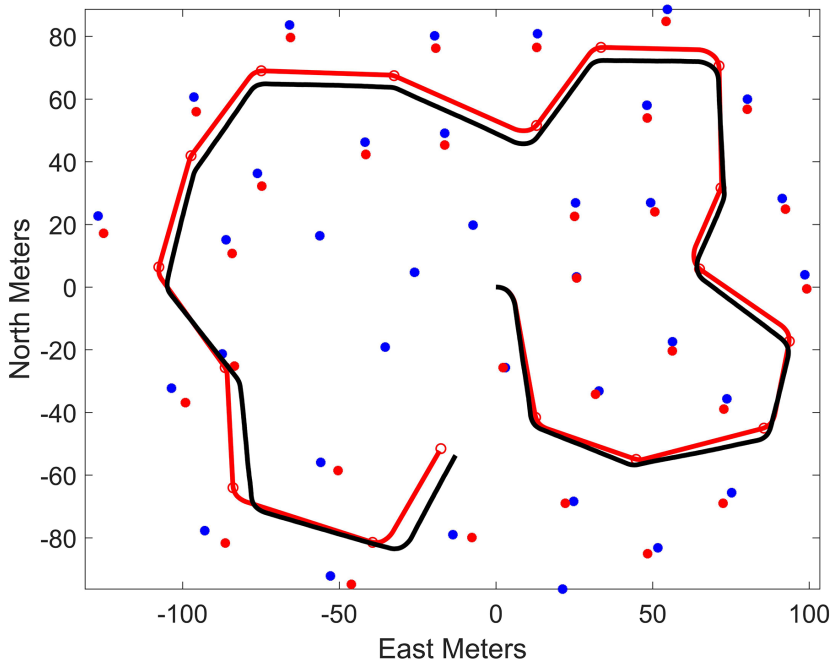
We use one hundred particles for simulation. Resampling is performed when $N_{eff} > 50$.

3.1.2. Results and Analysis

We simulated FastSLAM, PSO-FastSLAM (FastSLAM based on PSO with particle weights), AGA-FastSLAM (FastSLAM based on AGA-Resampling), and IFastSLAM with the same simulation parameters in order to verify the superiority of the improved algorithm. The estimated robot path and landmark location are shown in Figure 6.

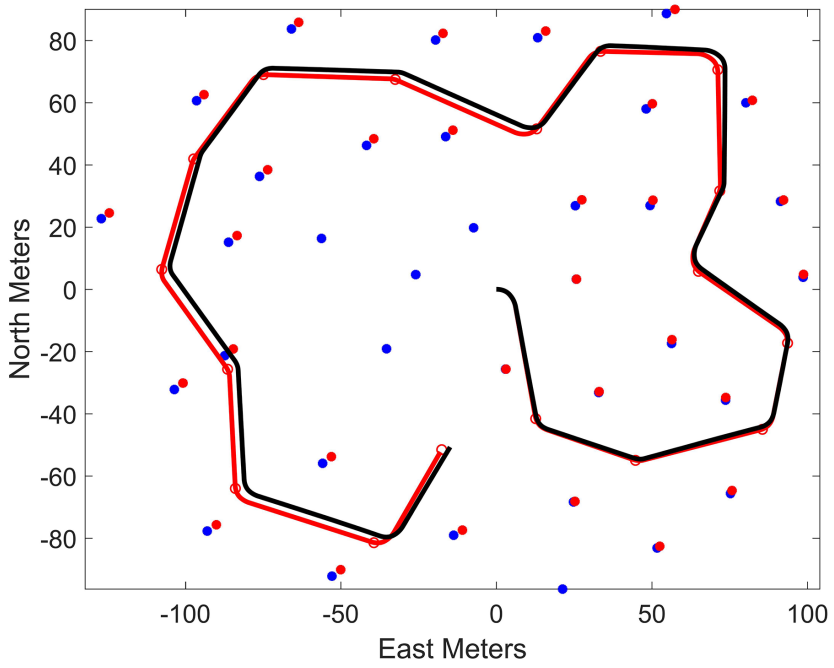


(**a**)

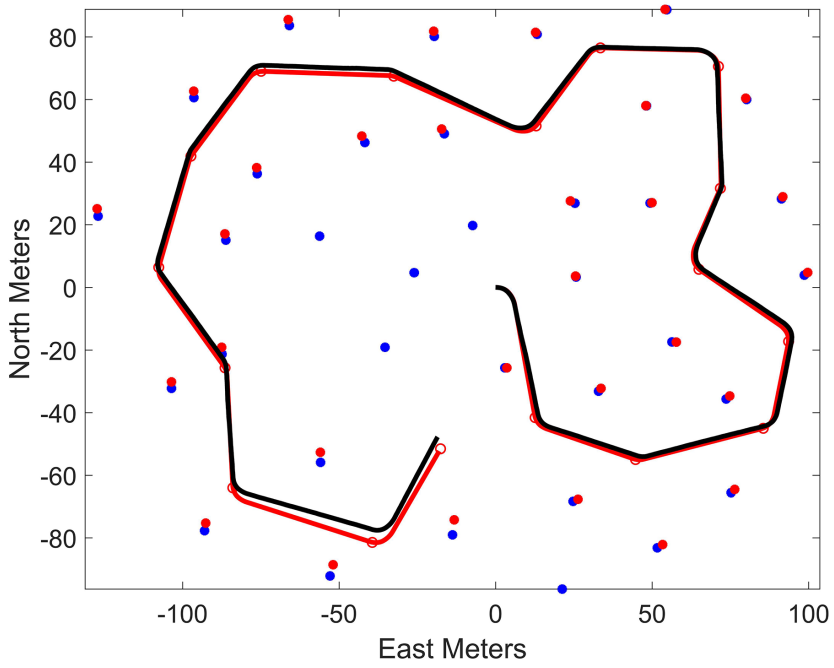


(**b**)

Figure 6. *Cont.*



(**c**)



(**d**)

Figure 6. The simulation result of four SLAM algorithms. (**a**) FastSLAM. (**b**) Particle swarm optimisation (PSO)-FastSLAM. (**c**) Adaptive generic FastSLAM algorithm (AGA-FastSLAM). (**d**) IFastSLAM.

From the results, we can conclude that the estimated landmark location of IFastSLAM is essentially consistent with the actual landmark location, and the estimated path substantially coincides with the actual path. The proposed IFastSLAM algorithm is therefore more accurate than the other three algorithms.

In order to compare these four algorithms more intuitively, we compared the location estimation error of the robot in X axis and Y axis, as shown in Figure 7. The positioning error is calculated as follows:

$$error(t) = \sqrt{(\sum_{i=1}^{N} x_i(t)/N - x_{true})^T (\sum_{i=1}^{N} x_i(t)/N - x_{true})} \quad (28)$$

The positioning error reflects the absolute error between the actual position of the robot and the average position of the particle swarm.

As shown in Figure 7a, the positioning error in X axis is less than 12 m in FastSLAM, less than 7 m in the standard PSO-FastSLAM algorithm optimised by particle weight, less than 4 m in AGA-FastSLAM, and within 3 m in IFastSLAM. The values of the estimation accuracy of the three improved FastSLAM algorithms are higher than that of the standard FastSLAM. In PSO-FastSLAM, the positioning accuracy is effectively improved due to the use of PSO optimisation. In AGA-FastSLAM, adaptive genetic resampling is used to handle the particle degeneracy problem, and the positioning accuracy is therefore also effectively improved. Furthermore, since the position of the virtual particle is used as the optimisation target, and the premature convergence phenomenon is solved in the FCPSO algorithm, the positioning error in IFastSLAM is smaller than in the other three algorithms. Similarly, in the IFastSLAM algorithm, the positioning error in Y axis is also smaller than for the other three algorithms, and the estimation accuracy is higher, as can be seen from Figure 7b.

As shown in Table 1, the IFastSLAM algorithm requires a shorter calculation time than PSO-FastSLAM and IFastSLAM for the same number of particles; the reason for this is that the use of fractional differential theory improves the evolution speed of the particle swarm. In addition, the calculation time increases as the particle number increases.

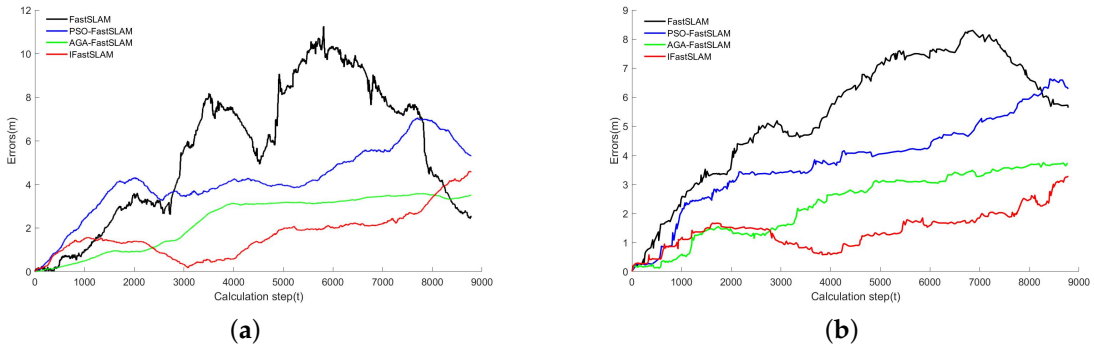


Figure 7. The estimation error. (**a**) Robot position error in X axis. (**b**) Robot position error in Y axis.

Table 1. Comparison of the RMSE and running time of SLAM algorithms.

Particle Number		**20**	**50**	**100**
FastSLAM	Time (s)	30.49	76.13	153.57
	RMSE (m)	8.43	7.90	6.84
PSO-FastSLAM	Time (s)	34.02	81.35	159.05
	RMSE (m)	6.31	5.10	4.79
AGA-FastSLAM	Time (s)	33.95	80.96	158.84
	RMSE (m)	5.12	3.78	3.25
IFastSLAM	Time (s)	31.17	78.05	156.98
	RMSE (m)	2.85	2.10	1.71

As shown in Figure 8, the estimation accuracy of IFastSLAM is significantly higher than the other three algorithms for the same number of particles. We can also clearly see that FCPSO-FastSLAM maintains a low RMSE with fewer particles than the other three algorithms, since the use of the virtual particle in the IFastSLAM algorithm, it can maintain more effective particles than the FastSLAM algorithm and can therefore resolve the problem of particle degeneracy to some extent.

Compared with standard FastSLAM, IFastSLAM sacrifices a certain amount of calculation time, but its estimation accuracy is significantly improved, and the running time is lower than for PSO-FastSLAM and AGA-FastSLAM. In this sense, the IFastSLAM has better overall performance.

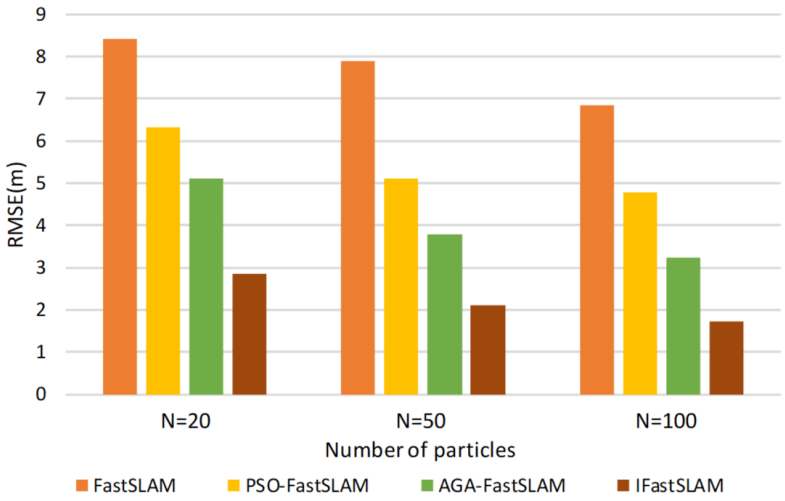


Figure 8. Simulation environment.

3.2. Verification Using the Victoria Park Dataset

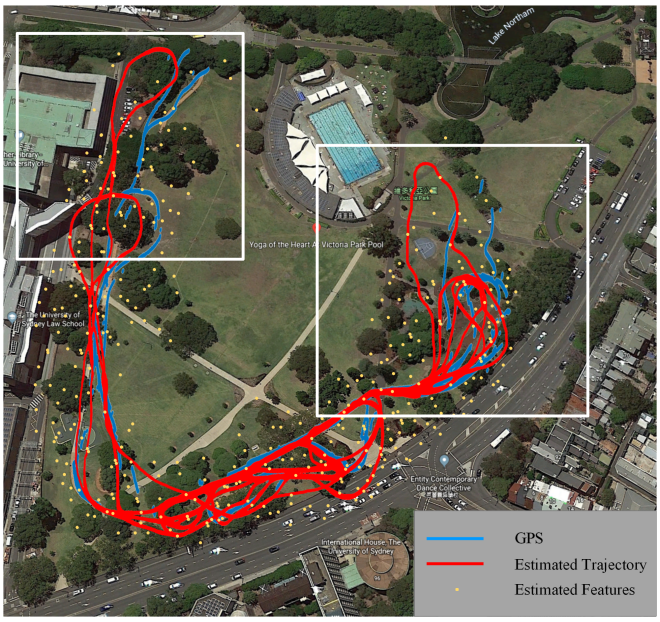
In order to further compare the performance of IFastSLAM with PSO-FastSLAM and AGA-FastSLAM, we applied several algorithms to the Victoria Park dataset [38]. This dataset is provided by the Australian Center for Field Robotics (ACFR) and is widely applied by researchers to verify the effectiveness and evaluate the performance of the SLAM algorithm in the real environment.

The mobile test platform used by the ACFR is shown as Figure 9. It is equipped with LiDAR, an odometer, and GPS, and is driven a distance of 4 km over about half an hour. LiDAR is used to provide observation information on the features of the road (trees in the park). The odometer is used to provide information on the linear speeds and heading angles of vehicles. The GPS sensor is used to provide reference vehicle location information. In this dataset, the errors caused by the inaccuracy of GPS data is ignored, and the vehicle trajectory is approximately represented by GPS data [39].

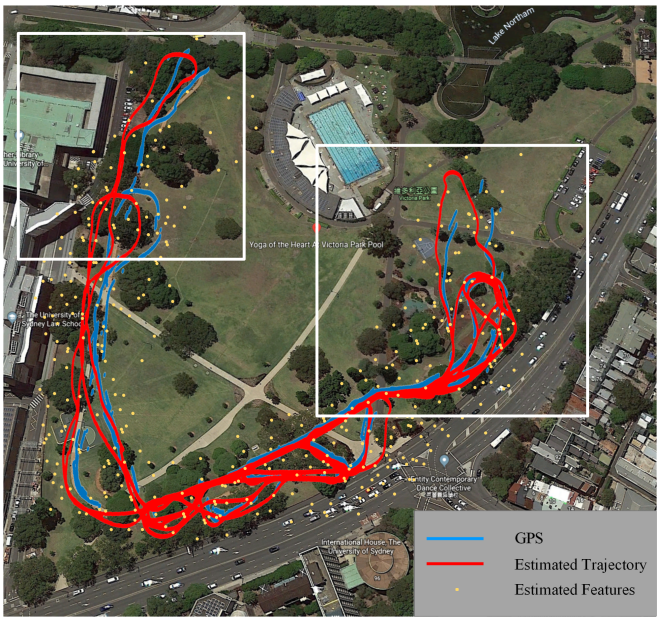
Figure 9. The mobile test platform used by the ACFR.

The validation results of several algorithms on the Victoria Park dataset are shown in Figure 10a–c, where the blue line represents the real GPS track of the vehicle, the red line represents the vehicle track estimated by the algorithm, and the yellow point represents the position of the landmark feature point. From the white box in Figure 10, it can be seen that the IFastSLAM algorithm obtains an estimation result that is closer to the real GPS track.

The RMSE of the robot trajectory and CPU running times of three algorithms on the Victoria Park dataset are shown in Table 2. We can draw the conclusion that the RMSE of IFastSLAM is lower than the other two algorithms, and it also requires less running time. Overall, we have verified the superior performance of IFastSLAM on the Victoria Park dataset.

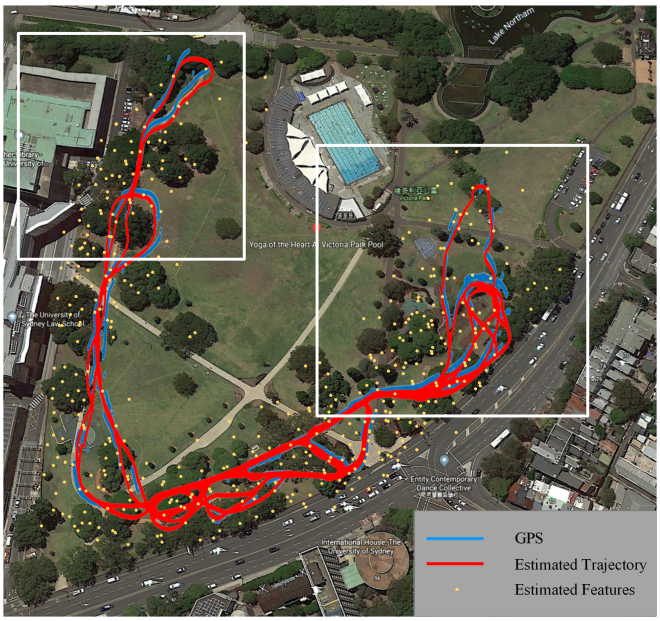


(**a**)



(**b**)

Figure 10. *Cont.*



(**c**)

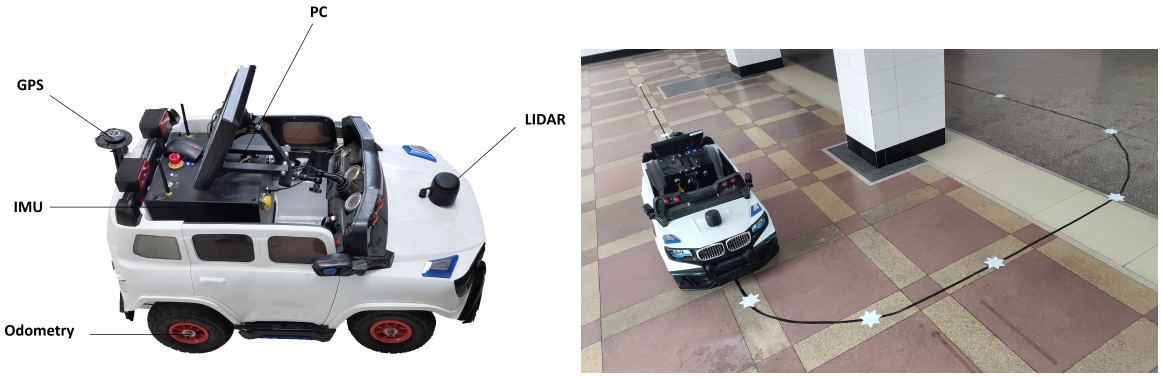
Figure 10. The Victoria Park dataset validation results. (**a**) PSO-FastSLAM. (**b**) AGA-FastSLAM. (**c**) IFastSLAM.

Table 2. Comparison of RMSE and running time with Voctoria Park dataset.

Algorithms	Running Time (s)	RMSE (m)
PSO-FastSLAM	1359.22	16.37
AGA-FastSLAM	1491.30	11.49
IFastSLAM	1406.57	9.15

3.3. Field Experiment

We used the mobile robot shown in Figure 11a to perform field experiments to test the practicability of IFastSLAM. The robot was equipped with an internal odometer and a SLAMTEC RPLIDAR A2 LiDAR, and moved at a speed of 0.3 m/s to create a grid map in real-time using LiDAR and odometer data. The National Key Laboratory of Virtual Simulation of Chang'an University was selected as the experimental site, as shown in Figure 11b.

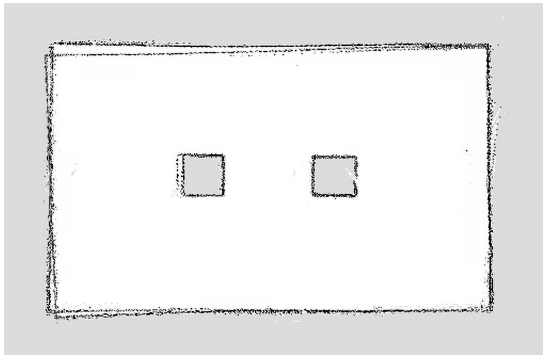


(**a**) (**b**)

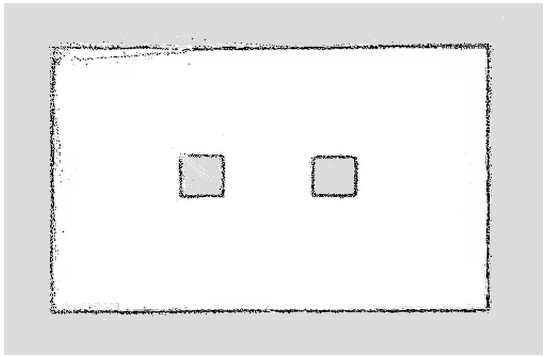
Figure 11. Field experiment condition. (**a**) Mobile robot used in the field experiment. (**b**) Field experimental site.

The robot was driven along the specified path within the experimental site to build an environmental map using radar scanning. Three algorithms were compared in this field experiment.

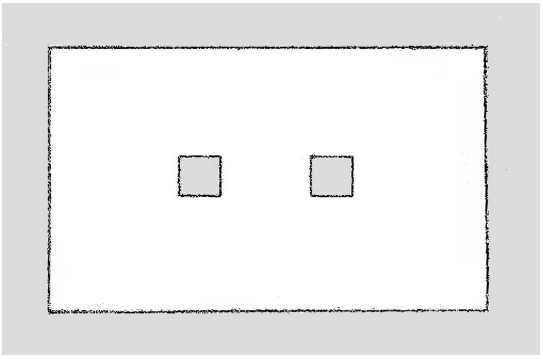
Figure 12a shows the map built by the PSO-FastSLAM algorithm, while Figure 12b,c show the maps built by AGA-FastSLAM and IFastSLAM in the same environment, respectively. There are obvious defects in the grid map built by PSO-FastSLAM as can be seen from Figure 12a, and a large deviation appears at the boundary of the map. Similarly, blurring occurs at the edge of the grid map built by the AGA-FastSLAM, as shown in Figure 12b. In contrast, the grid map built by the IFastSLAM is more precise, as shown in Figure 12c.



(**a**)



(**b**)



(**c**)

Figure 12. Grid map by different algorithm. (**a**) PSO-FastSLAM. (**b**) AGA-FastSLAM. (**c**) IFastSLAM.

The results reveal that the map generated by IFastSLAM has fewer defects and higher precision, while the maps generated by the PSO-FastSLAM and AGA-FastSLAM algorithms are of low quality.

Figure 13 shows the estimated trajectory for the different algorithms. We can see that the trajectory estimated by the IFastSLAM algorithm correctly follows the ground truth, while the trajectory estimated by PSO-FastSLAM and AGA-FastSLAM obviously deviates from the ground truth of the

robot in some places. IFastSLAM has better localisation accuracy than the other two algorithms. In addition, the running times of PSO-FastSLAM, AGA-FastSLAM, and IFastSLAM are 592.53 s, 647.09 s, and 629.74 s respectively, meaning that the computational efficiency of IFastSLAM is clearly higher. Table 3 show the RMSE of different algorithms in field experiment. It can be seen that the RMSE of IFastSLAM algorithm is the smallest. It can be therefore concluded that IFastSLAM has better overall performance.

Table 3. Comparison of RMSE for different algorithms in field experiment.

Algorithms	PSO-FastSLAM	AGA-FastSLAM	IFastSLAM
RMSE (m)	0.27	0.16	0.09

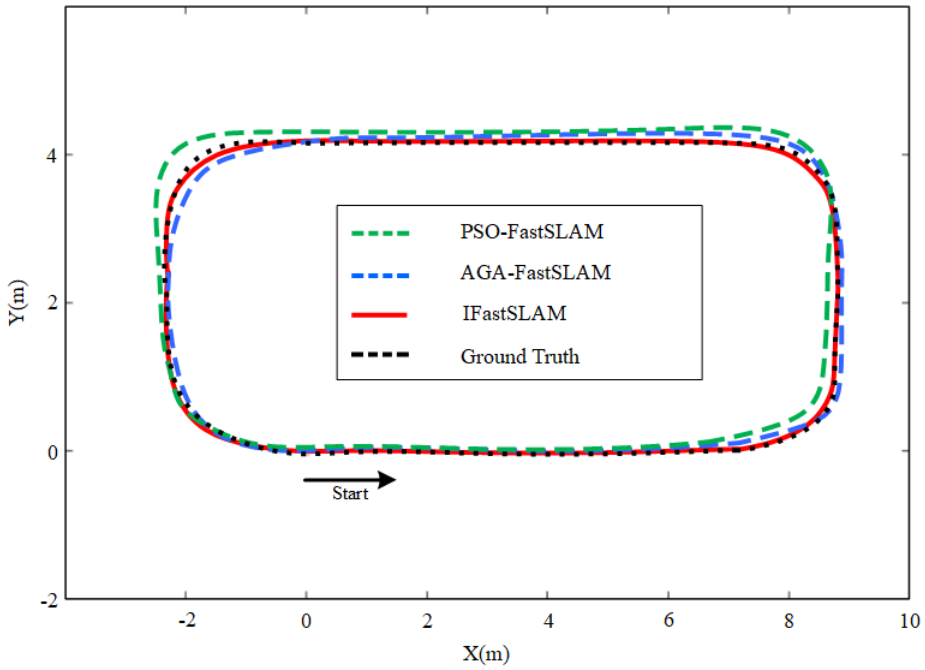


Figure 13. The estimated trajectory for different algorithms in the field experiment.

4. Conclusions

In the resampling process of the conventional FastSLAM process, the accuracy decreases with the number of iterations, due to particle degeneracy. We propose a framework named IFastSLAM that can enhance the performance of FastSLAM based on PSO. In this framework, an adaptive GA is introduced at the resampling stage, and a novel algorithm called FCPSO is proposed based on fractional differential and chaotic optimisation to improve PSO-FastSLAM. Compared with conventional frameworks, IFastSLAM has four significant benefits: (i) The diversity of particles is enriched by an adaptive GA; (ii) the evolutionary speed of the PSO algorithm is increase; (iii) the problem of premature convergence in PSO is alleviated; and (iv) the global optimisation accuracy is improved by using a virtual particle as the optimisation target, and the errors in the trajectory and landmark estimations are reduced.

We compare our proposed IFastSLAM approach with FastSLAM, PSO-FastSLAM, and AGA-FastSLAM on a series of SLAM issues. This comparison was made based on simulations, an experiment with the Victoria Park dataset provided by ACFR, and a field experiment with a mobile robot, showing promising results. In future work, we plan to integrate IFastSLAM into 3D maps and carry out further study on path planning and collision detection for mobile robots. In addition, the parameters used in FCPSO are taken directly from other studies, and we will readjust the parameters in the next study to make them more suitable for robot systems.

Author Contributions: Conceptualization, X.L. and B.F.; methodology, X.L. and B.F.; software, B.F.; validation, X.L., B.F. and W.L.; formal analysis, X.L., B.F. and W.L.; investigation, X.L., B.F., G.W. and Y.Y.; resources, X.L. and G.W.; data curation, B.F. and Y.Y.; writing—original draft preparation, X.L. and B.F.; writing—review and editing, G.W. and W.L.; visualization, B.F. and Y.Y.; supervision, G.W.; project administration, X.L. and G.W.; funding acquisition, X.L. and G.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Key Research and Development Programs of Shaanxi Province (grants 2019ZDLGY15-04-02, 2018ZDCXL-GY-05-04, 2018ZDCXL-GY-05-07-02) and the Fundamental Research Funds for the Central Universities (grants 300102329502, 300102320502).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this paper:

Symbols	Implication
SLAM	Simultaneous localization and mapping
FastSLAM	Rao-Blackwellized particle filter SLA
EKF-SLAM	extended Kalman filter SLAM
AGA-Resampling	adaptive genetic algorithm resampling
PSO	Particle Swarm Optimisation
FCPSO	fractional differential and chaotic PSO
IFastSLAM	FastSLAM based on AGA-Resampling and optimized by FCPSO
x_t	the state vector describing the position and orientation of the robot
u_t	the control vector, drive the robot from state x_{t-1} to state x_t
m_i	the vector describing the position of the ith landmark
z_t	the observation of the ith landmark from the vehicle at time t
$\omega_t^{(i)}$	the importance weight of ith particle
π	the proposal distribution of particles
N_{eff}	the effective number of particles
μ	the crossover degree of the particles
P_M	the mutation probability
P_i	the local optimal position of the particles of PSO
P_g	the global optimal position of the swarm of PSO
v_{id}^t	the velocity of ith particle at time t
x_{id}^t	the position of ith particle at time t
σ_f^2	the variance of particles' population fitness
f_i	the fitness of the ith particle
f_{avg}	the average fitness of the particle swarm
f	the normalization factor
m_x	the mean of the particle position
v	the moving speed of the robot
G	the steering angle of the robot
WB	the wheelbase of the robot
$W(t)$	the system noise at time t
$V(t)$	observation noise at time t
Q	systematic noise covariance
R	the observation noise covariance

Appendix A. Mathematical Proofs in the Paper

Theorem A1. *For a sequence in the range [0,1], the variance is in the range [0,1/4].*

Proof of Theorem A1. Assume that there is a sequence consists of n random variables in the range [0,1], namely X_1, X_2, ...,X_n. The average value of this sequence is denoted by Y. The variance is denoted by Z. According to the calculation formula of variance,

$$Z = [(X_1 - Y)^2 + (X_2 - Y)^2 + ... + (X_n - Y)^2]/n$$

$$= [(X_1)^2 + (X_2)^2 + ... + (X_n)^2 - 2Y(X_1 + X_2 + ... + X_n) + nY^2]/n$$
$$= [(X_1)^2 + (X_2)^2 + ... + (X_n)^2 - 2Y(nY) + nY^2]/n$$
$$= [(X_1)^2 + (X_2)^2 + ... + (X_n)^2 - nY^2]/n$$
$$= [n(X_1)^2 + n(X_2)^2 + ... + n(X_n)^2 - (X_1 + X_2 + ... + X_n)^2]/n^2 \quad \square$$

When X_2, X_3, ...,X_n are regarded as fixed values, the above formula is a quadratic function of opening upward for X_1. So when X_1 = 0 or X_1 = 1 (one of the two endpoints of the domain), the variance Z takes the maximum value.

Similarly, when X_2 = 0 or X_2 = 1, the variance Z takes the maximum value.

...

Similarly, when X_n = 0 or X_n = 1, the variance Z takes the maximum value.

Therefore, only when the values of all random variables take values in 0 or 1, the variance Z takes the maximum value. We will seek the maximum variance next. Assuming that in the sequence, the number of random variables with a value of 0 is a, and the number of random variables with a value of 1 is $(n - a)$, then the average Y= $(n - a)/n$.

$$\text{Z}=[a(0 - Y)^2 + (n - a)(1 - Y)^2]/n$$
$$=a[0 - (n - a)/n]^2/n + (n - a)[1 - (n - a)/n]^2/n$$
$$=[a(n - a)^2 + (n - a)a^2]/n^3$$
$$=a(n - a)/n^2$$
$$=(-a^2 + na)/n^2$$
$$=[-(a - n/2)^2 + n^2/4]/n^2$$
$$\leq 1/4$$

Therefore, when $a = n/2$, in other words, the number of random variables with a value of 0 is $n/2$, and the number of random variables with a value of 1 is $n/2$. Under these circumstances, the variance Z takes the maximum value of $1/4$. Because Z is a positive number, the variance Z is therefore in the range [0,1/4].

In conclusion, for a sequence in the range [0,1], the variance is in the range [0,1/4].

References

1. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [CrossRef]
2. Dissanayake, M.G.; Newman, P.; Clark, S.; Durrant-Whyte, H.F.; Csorba, M. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robot. Autom.* **2001**, *17*, 229–241. [CrossRef]
3. Maybeck, P.S. Stochastic models, estimation, and control, ser. *Math. Sci. Eng.* **1979**, *141*, 1.
4. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *AAAI/IAAI* **2002**, 593–598, doi:10.1007/s00244-005-7058-x. [CrossRef]
5. Murphy, K.P. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems*; The MIT Press: Denver, CO, USA, 2000; pp. 1015–1021.
6. Thrun, S.; Burgard, W.; Fox, D. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 1, pp. 321–328.
7. Bailey, T.; Nieto, J.; Nebot, E. Consistency of the FastSLAM algorithm. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; pp. 424–429.
8. Cugliari, M.; Martinelli, F. A FastSLAM algorithm based on the unscented filtering with adaptive selective resampling. In *Field and Service Robotics*; Springer: Berlin, Germany, 2008; pp. 359–368.
9. Liu, D.; Duan, J.; Yu, H. FastSLAM algorithm based on adaptive fading extended Kalman filter. *Syst. Eng. Electron.* **2017**, *38*, 644–651.
10. Zhang, Y.F.; Zhou, Q.X.; Zhang, J.Z.; Jiang, Y.; Wang, K. A FastSLAM algorithm based on nonlinear adaptive square root unscented Kalman filter. *Math. Probl. Eng.* **2017**, *2017*, 4197635. [CrossRef]

11. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [CrossRef]
12. Mallik, S.; Mallik, K.; Barman, A.; Maiti, D.; Biswas, S.K.; Deb, N.K.; Basu, S. Efficiency and cost optimized design of an induction motor using genetic algorithm. *IEEE Trans. Ind. Electron.* **2017**, *64*, 9854–9863. [CrossRef]
13. Li, T.; Bolic, M.; Djuric, P.M. Resampling methods for particle filtering: classification, implementation, and strategies. *IEEE Signal Process. Mag.* **2015**, *32*, 70–86. [CrossRef]
14. Lv, T.Z.; Zhao, C.X.; Zhang, H.F. An improved FastSLAM algorithm based on revised genetic resampling and SR-UPF. *Int. J. Autom. Comput.* **2018**, *15*, 325–334. [CrossRef]
15. Khairuddin, A.R.; Talib, M.S.; Haron, H.; Abdullah, M.Y.C. GA-PSO-FASTSLAM: A hybrid optimization approach in improving FastSLAM performance. In *International Conference on Intelligent Systems Design and Applications*; Springer: Berlin, Germany, 2016; pp. 57–66.
16. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995.
17. Kwok, N.M.; Liu, D.; Dissanayake, G. Evolutionary computing based mobile robot localization. *Eng. Appl. Artif. Intell.* **2006**, *19*, 857–868. [CrossRef]
18. Todor, B.; Darabos, D. Simultaneous localization and mapping with particle swarm localization. In Proceedings of the 2005 IEEE Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Sofia, Bulgaria, 5–7 September 2005; pp. 216–221.
19. Havangi, R.; Taghirad, H.D.; Nekoui, M. A.; Teshnehlab, M. A square root unscented FastSLAM with improved proposal distribution and resampling. *IEEE Trans. Ind. Electron.* **2013**, *61*, 2334–2345. [CrossRef]
20. Zhao, Y.; Wang, T.; Qin, W.; Zhang, X. Improved Rao-Blackwellised particle filter based on randomly weighted particle swarm optimization. *Comput. Electr. Eng.* **2018**, *71*, 477–484. [CrossRef]
21. Lee, S.H.; Eoh, G.; Lee, B.H. Relational FastSLAM: An improved Rao-Blackwellized particle filtering framework using particle swarm characteristics. *Robotica* **2016**, *34*, 1282–1296. [CrossRef]
22. Zuo, T.; Min, H.; Tang, Q.; Tao, Q. A Robot SLAM Improved by Quantum-Behaved Particles Swarm Optimization. *Math. Probl. Eng.* **2018**, *2018*. [CrossRef]
23. Thrun, S. Probabilistic robotics. *Commun. ACM* **2002**, *45*, 52–57. [CrossRef]
24. Kwak, N.; Lee, B.H.; Yokoi, K. Result representation of Rao-Blackwellized particle filtering for SLAM. In Proceedings of the 2008 International Conference on Control, Automation and Systems, Seoul, Korea, 14–17 October 2008; pp. 698–703.
25. Yin, S.; Zhu, X. Intelligent particle filter and its application to fault detection of nonlinear system. *IEEE Trans. Ind. Electron.* **2015**, *62*, 3852–3861. [CrossRef]
26. DeGroot, M.H.; Schervish, M.J. *Probability and Statistics*; Pearson Education: NewYork, USA, 2012.
27. Back, T. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In Proceedings of the 2nd Conference of Parallel Problem Solving from Nature, Brussels, Belgium, 28–30 September 1992; Elsevier Science Publishers: Amsterdam, The Netherlands, 1992.
28. Zhang, X.; Zou, D.; Shen, X. A Novel Simple Particle Swarm Optimisation Algorithm for Global Optimisation. *Mathematics* **2018**, *6*, 287. doi:10.3390/math6120287. [CrossRef]
29. Couceiro, M.S.; Ferreira, N.F.; Machado, J.T. Application of fractional algorithms in the control of a robotic bird. *Commun. Nonlinear Sci. Numer. Simul.* **2010**, *15*, 895–910. [CrossRef]
30. Pires, E.S.; Machado, J.T.; de Moura Oliveira, P.; Cunha, J.B.; Mendes, L. Particle swarm optimization with fractional-order velocity. *Nonlinear Dyn.* **2010**, *61*, 295–301. [CrossRef]
31. Couceiro, M.S.; Rocha, R.P.; Ferreira, N.F.; Machado, J.T. Introducing the fractional-order Darwinian PSO. *Signal Image Video Process.* **2012**, *6*, 343–350. [CrossRef]
32. Couceiro, M.S.; Ferreira, N.; Tenreiro Machado, J. Fractional order Darwinian particle swarm optimization. In *Symposium on Fractional Signals and Systems*; Springer International Publishing, Berlin, Germany, 2011; pp. 127–136.
33. Fan, S.K.S.; Jen, C.H. An Enhanced Partial Search to Particle Swarm Optimization for Unconstrained Optimization. *Mathematics* **2019**, *7*, 357. [CrossRef]

34. Pan, I.; Korre, A.; Das, S.; Durucan, S. Chaos suppression in a fractional order financial system using intelligent regrouping PSO based fractional fuzzy control policy in the presence of fractional Gaussian noise. *Nonlinear Dyn.* **2012**, *70*, 2445–2461. [CrossRef]
35. Shi, Y.; Chen, G. Chaos of discrete dynamical systems in complete metric spaces. *Chaos Solitons Fractals* **2004**, *22*, 555–571. [CrossRef]
36. Zhang, G.; Cheng, Y.; Yang, F.; Pan, Q. Particle filter based on PSO. In Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA), Hunan, China, 20–22 October 2008; Volume 1, pp. 121–124.
37. Bailey, T. Source code for SLAM simulations of Tim Bailey. Available online: www.acfr.usyd.edu.au/homepages/academic/tbailey/software (accessed on 1 March 2020).
38. ACFR. Victoria Park Dataset. Available online: www.acfr.us-yd.edu.au/homepages/academic/enebot/dataset.htm. (accessed on 1 March 2020).
39. Nieto, J.I.; Guivant, J.E.; Nebot, E.M.; Thrun, S. Real time data association for FastSLAM. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 14–19 September 2003; Volume 1, pp. 412–418.