International Journal of Engineering

Journal Homepage: www.ije.ir

Improved Simultaneous Localization and Mapping Estimation using Crow Search Algorithm Based Particle Filter

M. Abedini, H. Jazayeriy*, J. Kazemitabar

Faculty of Electrical & Computer engineering, Noshirvani University of Technology, Babol, Iran

PAPER INFO

Paper history:
Received 05 June 2023
Received in revised 27 June 2023
Accepted 28 June 2023

Keywords:
Particle Filter
Fast Simultaneous Localization and Mapping
Crow Search Algorithm
Sample Depletion

ABSTRACT

Trajectory tracking and positioning are essential requirements in many areas, including robots and autonomous vehicles. In some cases, such as in areas where GPS signals are weak or not available, trajectory tracking is used as an alternative positioning system. In these cases, simultaneous localization and mapping (SLAM), is of great importance as it does not require prior knowledge and empirical offline fingerprint. SLAM can be combined with signal processing algorithms among which, particle filter stands out. However, challenges exist such as particle weights degradation and particles impoverishment that need to be dealt with. In fact, the loss of particle diversity for estimation has led to the lack of particles. To overcome this problem, one solution is to diversify the selection of particles after resampling. In this paper, we proposed a crow search algorithm (CSA) to overcome these issues and improve position estimation. The simulation results showed that this algorithm greatly improved the performance of fast SLAM.

***doi**: 10.5829/ije.2023.36.10a.09*

NOMENCLATURE

CSA	Crow Search Algorithm	$x^{i,it}$	Position of the crow i in iteration it
x_k	The state of the moving vehicle or mobile robot in the time k	AP	Awareness Probability
z_k	Observation in the time k	fl	Flight length
x_k^i	The state of the particle i in the time k	F	Fitness function
w_k^i	The weight of the particle i in the time k	v_c	Robobt velocity
q	Importance density	(x_v, y_v)	2-D Position of the robot
l	Landmark position	(x_l, y_l)	2-D landmark position
$m^{i,it}$	Position of the crow's hide-out i in the iteration it	**Greek Symbols**	
it	Number of iterations	ϕ	Robot orientation

1. INTRODUCTION

Simultaneous localization and mapping techniques have been around for many years in a variety of areas, such as human positioning, unmanned vehicles, submarines, and robots, including those used inside the human body. When SLAM was proposed nearly two decades ago it was initially viewed as a side idea. Nowadays, however, it is considered as an inexpensive and integral component of robots and automated moving objects technology. The SLAM problem can be categorized according to different factors. The SLAM technique used in robots first introduced by Smith et al. [1] which was used to construct an unknown environment map, while simultaneously determining its location. In SLAM, unlike fingerprinting methods [2], both path and position are estimated online without the need for any prior knowledge of the environment. Different SLAM methods, despite their core similarities, are distinguished based on the way sensors are used and implemented. Kalman filters are one of the first techniques to solve SLAM. Extended Kalman Filter (EKF) [3] is used to estimate the state and position of environmental signals on a robot. Given controls $u_{0:k} = \{u_0, \cdots, u_k\}$ and robot observations $z_{1:k} =$

*Corresponding Author Institutional Email: Jhamid@nit.ac.ir (H. Jazayeriy)

$\{z_1, \cdots, z_k\}$, we look for the landmarks location or map (l) and the pose $x_{0:k} = \{x_0, x_1, \cdots, x_k\}$. In other words, Kalman filter provides a solution to the online SLAM problem, $p(x_k, l | u_{0:k}, z_{1:k})$. The EKF estimation has an error because it uses the Taylor series approximation of the nonlinear estimation function. Also, the EKF estimation is not of much help when the model is both non-linear and non-Gaussian. Therefore, solving the SLAM problem using the Monte Carlo method was one of the other methods proposed to overcome these challenges [4]. Instead of calculating the integral at all points in the Bayesian formulation, it does so by sampling at the points that have the greatest contribution to the integral calculation. Points sampling in Monte Carlo is performed using Perfect Monte Carlo, Rejection Sampling (RS), Importance Sampling (IS), and Sequential Importance Sampling (SIS) methods [5]. The idea behind the SIS method, which is a special case of IS, is to re-use the samples generated in the previous steps to sample the posterior distribution function in the new step. The challenge facing the SIS is weight degeneracy, where the variance of weights for the proposed distribution increases with each step. To overcome this, re-sampling can be used to reduce the variance of weights. The so-called resampling in SIS is called sequential importance resampling (SIR) or particle filter [6]. One of the methods of applying a particle filter to solve the SLAM problem is known as Fast SLAM. Fast SLAM breaks down a SLAM problem into a robot positioning problem and a set of landmark estimation problems that are conditional on the robot status estimation [7]. So far, advanced versions of Fast SLAM have been offered by Lei et al. [8], but all of them are based on one basic rule; as reported by Murphy [9], the representation as such is accurate due to the natural conditional independence in the SLAM problem. Fast SLAM uses a modified particle filter to estimate the posterior paths of the robot. Each particle has k Kalman filters that estimate the K positions of the landmark based on the path estimation. The resulting algorithm is an example of a Rao-Blackwellized particle filter [10, 11].

With the rapid development of computer hardware performance, particle filter is applied in SLAM [12]. The problem with the traditional particle filtering algorithm is the contrast between the degradation and the lack of particles. Re-sampling resolves the problem of particle degradation to some extent, but creates another problem known as sample depletion. In addition, high-precision estimation requires the application of large numbers of particles, which results in computational complexity and inconsistency.

There are two conventional ways to overcome this problem. The first is to diversify the choice of particle location and the other is to better allocate weight to the particles. In this paper, we select the first solution to overcome this problem. To accomplish this method, scholars have conducted a great deal of studies and have reached the important point that particle filtering based on intelligent optimization mechanism of biological group is a new perspective [13, 14]. In fact, in these methods, the particles in the particle filter are considered as individuals in biological clusters, and the particle distribution using the simulated biological cluster motion law is more reasonable.

Huang et al. [15], and Chen et al. [16], cuckoo and bat algorithms were introduced to have better performance of the particle filtering algorithm. Gao et al. [17] used the firefly algorithm transfer formula to recombine the particle sample. However, re-sampling is still required in this method and optimization of the duplicate particles is not performed. Tian et al. [18] optimized the particle filter using the firefly algorithm, which shows that the firefly algorithm can update particle states in a way that prevents particle starvation. Another advantage of this method is that the same computational efficiency can be achieved with fewer particles.

Although these methods and other methods such as the use of genetic algorithms in target tracking and trajectory tracking control have been proposed [19–21], the issue of simultaneous mapping, in addition to localization, plays a very important role in robots and unmanned vehicles. Zhu et al. [12], combined the particle filtering mechanism in SLAM with improved firefly lighting formula. Moreover, they used the dynamic equilibrium algorithm which delivers global and local simultaneously.

Following new research on particle filtering that harnesses optimization mechanisms of intelligent bio-groups [22, 23], these researches used CSA algorithm. Our goal is to merge CSA and particle filtering to improve resampling of particles, optimize particle weights and obtain higher accuracy with smaller number of particles. We term this method CSAPF-SLAM.

In fact, by creating a reasonable variation in the position of the particles using mechanisms of intelligent bio-groups, we save them from getting involved in local extremes, and thus achieve a better estimate. It seems that the use of metaheuristic algorithm is efficient to diversify the selection of particles after resampling and among them we have chosen the crow search algorithm. Crow search algorithm is one of the MA algorithm that attracted much attention from researchers since it was introduced. The evaluation results of CSA show that it is very efficient for optimization problems, especially problems that science and engineering have difficulty to solve [24]. Meanwhile, this algorithm is easy to implement and has only a few parameters [25]. Also, by combining the particle filter and the crow search algorithm, it is possible to propose a fitness function that solves some problems such as entrapment in the local optimum due to the AP parameter. Furthermore, the algorithm uses particle filter to keep the balance between the local and global search processes. In fact, a novel neighborhood assigning strategy has been introduced to optimize the local search.

Short statement of precise problem addressed in this manuscript:

- A new method for more accurate estimation of simultaneous localization and mapping in robots in non-linear and non-Gaussian condition.
- A new algorithm to overcome the impoverishment problem in particle filter using CSA.
- A new fitness function for SLAM using particle filter.

In section II, we describe particle filter SLAM and CSA algorithm. Section III describes our proposed method and section IV demonstrates experimental valuations based on Sydney University Dataset [26, 27]. Finally, section V concludes the paper.

2. PARTICLE FILTER AND CSA ALGORITHM

Since the proposed method is a combination of particle filter and crow search algorithm, we describe each of these, separately. We will then explain our proposed algorithm, i.e. CSAPF-SLAM.

2. 1. Particle Filter SLAM Algorithm

As we know, in SLAM, we are dealing with two issues of localization and mapping [28]. The following equation is used for the problem of moving vehicle or mobile robot position in filter based SLAM algorithms:

$$\tilde{x}_k = \int x_k \, p(x_k|z_{1:k}) dx_k \tag{1}$$

where x_k is the state of the moving vehicle or mobile robot in the time k and z_k is the observation in the time k.

Filter-based localization methods include the following two steps:

A) Prediction step (time update): The Prediction step according to Chapman-Kolmogorov principle can be as follows:

$$p(x_k|u_{0:k}, z_{1:k-1}, x_0) = \int p(x_k|x_{k-1}, u_k) \times p(x_{k-1}|u_{0:k-1}, z_{1:k-1}, x_0) dx_{k-1} \tag{2}$$

B) Correction step (measurement update): In time k, a value of z_k is available and this can be used for the previous update (time update) via the Bayesian rule:

$$p(x_k|u_{0:k}, z_{1:k}, x_0) = \frac{p(z_k|x_k \quad)p(x_k|u_{0:k}, z_{1:k-1}, x_0)}{p(z_k|u_{0:k}, z_{1:k-1} \,)} \tag{3}$$

To overcome the limitation caused by the Gaussian condition of the distributions as well as the linearization error in EKF, the use of a particle filter is suggested by Arulampalam et al. [29]. In this case, the term $p(x_k|z_{1:k})$ in Equation (1) is transformed into the following:

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \cdot \delta(x_k - x_k^i) \tag{4}$$

where x_k^i is the state of the particle i in the time k and w_k^i is the weight of the particle i in the time k, where particle weight is determined by the following equations:

$$w_k^i \propto \frac{p(x_{0:k}^i|z_{1:k})}{q(x_{0:k}^i|z_{1:k})} \tag{5}$$

where q is the importance density. Since it is common that only the filtered estimate of $p(x_k|z_{1:k})$ is required at each time step, we have the following relation by performing a series of algebraic operations:

$$w_k^i \propto w_{k-1}^i \cdot \frac{p(z_k|x_k^i) \cdot p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)} \tag{6}$$

Considering the sequential importance resampling (SIR), which includes selecting the previous density $p(x_k|x_{k-1})$ as the importance density $q(x_k \,|x_{1:k-1}, z_k)$, we will have the following recursive equation:

$$w_k^i \propto w_{k-1}^i \cdot p(z_k|x_k^i) \tag{7}$$

Finally, given the location and weight of each particle, the position of the robot or vehicle can be estimated according to the following equation:

$$\underline{\tilde{x}}_k = \sum_{j=1}^{N_s} w_k^j \cdot x_k^j \tag{8}$$

```
1: for $i = 1{:}N_s$
  2:     Randomly choose initial state $x_0^i$ in the search space
  3:     $w_0^i = 1/N_s$
4: end for
5:$k$=1
6:while ($k <> 0$)
7:     // New pose selection and update weights with new observations
8:     for $i = 1{:}N_s$
9:         sample a new pose $x_k^i \sim p(x_k|x_{k-1}^i, u_k)$
10:        $w_k^i = w_{k-1}^i \cdot p(z_k|x_k^i)$
11:    end for
12: Apply the selected resampling method for the set of particles
        and their weights to get a new set of particles and weights
13:  // Calculate the estimate (Same as probability integral from
         probability density function)
14:    for $j = 1{:}N_s$
15:        $\underline{\tilde{x}}_k = w_k^j \cdot x_k^j$
16:        $x_k$+= $\underline{\tilde{x}}_k$
17:    end for
18:    $k = k + 1$
  19:    if fitness function condition is realized
  20:    $k = 0$
  21:    end if
22:end while
```

Figure 1. Pseudo code for particle filter algorithm

In general, Figure 1 shows the pseudo code for particle filter algorithm for estimating the state. It should be noted that the fitness function is calculated based on the importance weight.

So far, we have only discussed positioning in the SLAM algorithm, but mapping should also be performed in SLAM. In other words, we are dealing with the posterior probability density $p(x_k,l|u_{0:k},z_{1:k},x_0)$. For our proposed algorithm, we will use one of the most efficient methods using a particle filter called Fast SLAM. As mentioned in the introduction, Fast SLAM uses only the particle filter according to the above equations to determine the position and uses a number of EKF for mapping. Therefore here, according to the posterior factorization provided by Murphy [9], the SLAM posterior will be as follows:

$$p(x_{1:k},l_{1:m}|z_{1:k},u_{0:k-1}) = p(x_{1:k}|z_{1:k},u_{0:k-1}) \cdot p(l_{1:m}|x_{1:k},z_{1:k}) \tag{9}$$

where $\mathrm{p}(\mathrm{x}_{1:k}|\mathrm{z}_{1:k},\mathrm{u}_{0:k-1})$ is the posterior density of the robot state and $\mathrm{p}(\mathrm{l}_{1:m}|\mathrm{x}_{1:k},\mathrm{z}_{1:k})$ is the position of the landmarks, which we have due to the conditional independence in the position of landmarks:

$$p(x_{1:k},l_{1:m}|z_{1:k},u_{0:k-1}) = p(x_{1:k}|z_{1:k},u_{0:k-1}) \cdot \prod_{i=1}^{M} p(l_i|x_{1:k},z_{1:k}) \tag{10}$$

The first term on the right side of the above equation is obtained by using the particle filter in accordance with Equation (3). In other words, for each particle in each step, after determining the position, the position of the landmarks must also be estimated. That is, each particle has a memory of $2m + 3$, in which m is the number of landmarks. Figure 2 will help us understand this concept. In this figure, μ_k^i and Σ_k^i represent mean and covariance of particle i at time k, respectively. Figure 3 displays the pseudo code for Fast SLAM.

2. 2. Crow Search Optimization Algorithm (CSA)

Crow search meta-heuristic is a relatively new algorithm introduced in 2016 inspired by Crows' social and intelligent behaviour by Askarzadeh [30]. Crows (family or species of crows) are recognized as one of the cleverest birds in nature. It can be said that after humans, crows have proportionally the largest brain among living beings, and this is the reason for the intelligent behaviour of these birds [30]. In fact, this is proven by the brain to body ratio. Reasons to consider crows as smart birds are summarized as follows: Ability to remember human faces, conspiracy capability, powerful memory, ability to solve problems and planning. Crows can remember faces and are able to alert an anonymous face to one another. They also have self-awareness in the mirror and are also able to make and use of tools, as well as communicate in sophisticated ways, or remember to hide their food for several months. Unknown communication systems exist among crows. One of them is the complex language between crows. Crows also stock their surplus food. It has been widely seen that other crows have found their food supply in pursuit of another. Because crows are cautious and masterful in hiding, researchers believe that crows cannot rob another food crop without planning. They are also highly capable of hiding and storing food. In fact, crows can predict the behaviour of thieves using their experience of theft and can determine the safest path to protect their warehouses from theft [30].

The CSA algorithm is such that, like all community-based algorithms, it assumes N as the number of crows in the d-dimensional search space. Each crow i is defined by the vector in this search space as follows:

$$x^{i,it} = [x_1^{i,it}, x_2^{i,it}, \cdots, x_d^{i,it}] \tag{11}$$

where $i = 1,2,\dots,N$, $it = 1,2,\dots,it_{max}$ and it_{max} is the largest number of iterations. Each crow also has a memory that remembers the best experience of the hide-out location. In each iteration the position of the crow's hide-out i is indicated by $m^{i,it}$. This is the best location Crow i has ever seen and is described as a vector:

$$m^{i,it} = [m_1^{i,it}, m_2^{i,it}, \cdots, m_d^{i,it}] \tag{12}$$

Figure 2. Memory diagram of each particle

Particle (*i*)	**Vehicle or robot path at time *k***	**Mean and covariance of feature 1**	**Mean and covariance of feature 2**	**…**	**Mean and covariance of feature *m***
1	$x_{1:k}^1 = \{(x \;\; y \;\; \theta)^T\}_{1:k}^1$	$\mu_1^1 \; \Sigma_1^1$	$\mu_2^1 \; \Sigma_2^1$	…	$\mu_m^1 \; \Sigma_m^1$
2	$x_{1:k}^2 = \{(x \;\; y \;\; \theta)^T\}_{1:k}^2$	$\mu_1^2 \; \Sigma_1^2$	$\mu_2^2 \; \Sigma_2^2$	…	$\mu_m^2 \; \Sigma_m^2$
.	.	.	.	.	.
.	.	.	.	…	.
.	.	.	.	.	.
N_s	$x_{1:k}^{N_s} = \{(x \;\; y \;\; \theta)^T\}_{1:k}^{N_s}$	$\mu_1^{N_s} \; \Sigma_1^{N_s}$	$\mu_2^{N_s} \; \Sigma_2^{N_s}$	…	$\mu_m^{N_s} \; \Sigma_m^{N_s}$

```
1:for $i = 1{:}N_s$
2:      Randomly choose initial state $x_0^i$ in the search space
3:      $w_0^i = 1/N_s$
4:end for
5:$k$=1
6:while ($k <> 0$)
7:   // New pose selection and update weights with new
        observations
8:      for $i = 1{:}N_s$
9:          sample a new pose $x_k^i \sim p(x_k|x_{k-1}^i, u_k)$
10:         $w_k^i = w_{k-1}^i \cdot p(z_k|x_k^i)$
11:       // Incorporate the measurement $z_k^i$ into the corresponding
            EKF for mapping
12:           for $j = 1{:}m$
13:               update mean $\mu_{j,k}^i$
14:               update covariance $\Sigma_{j,k}^i$
15:           end for
16:     end for
17:         Apply the selected resampling method for the set of
        particles and their weights to get a new set of particles and
        weights
18:   // Calculate the estimate (Same as probability integral from
          probability density function)
19:     for $j = 1{:}N_s$
20:         $\underline{\tilde{x}}_k = w_k^j . x_k^j$
21:         $x_k += \underline{\tilde{x}}_k$
22:         Estimate map using $\mu_k^j$ and $\Sigma_k^j$
23:     end for
24:     $k = k + 1$
25:     if fitness function condition is realized
26:     $k = 0$
27:     end if
28:end while
```

Figure 3. Pseudo code for Fast SLAM

Crows move around in search of better food sources or hide-outs. The steps for updating the crow's position are as follows:

Step One: A crow, for example j, is randomly selected from the population. The crow i tries to follow the crow j to find his hideout (m_j). In this case, two modes can be created based on the crow j's knowledge of the chase by i. A random number is generated with a uniform distribution between 0 and 1. If this random value is greater than a pre-defined parameter named as Awareness Probability (AP) we go to step 2, otherwise we go step 3.

Step Two: In this step the crow j does not know that crow i is chasing him. Thus, crow i, according to the following relationship, reaches the crow's hide-out j:

$$x^{i,it+1} = x^{i,it} + r_i \times fl \times \left(m^{i,it} - x^{i,it}\right) \tag{13}$$

where $x^{i,it+1}$ is the location of crow i in iteration $t+1$ and $x^{i,it}$ is the position of the crow i in iteration t and $m^{i,it}$ is the position of the crow j in iteration t and r_i is the random number described earlier and fl is called flight length. Figure 4 schematically illustrates this case and shows the effect of fl on the search ability. Large values are the result of global search (far) and small values of fl are the result of local (neighbourhood) search. As shown in Figure 4, the next position of the crow i is on the line between $x^{i,it}$ and $m^{i,it}$. If fl is less than 1, the next position of the crow i is on a line that may exceed $m^{i,it}$ if fl is greater than 1.

Step Three: Crow j knows that crow i is chasing him. Therefore, to protect his stockpile from theft, crow j moves randomly to another location (search space) and the crow i will be fooled. This can be summarized in what follows:

$$x^{i.it+1} = \begin{cases} x^{i,it} + r_i \times fl \times \left(m^{i,it} - x^{i,it}\right) & if\ \ r_i \geq AP \\ \ \ a\ random\ position, & otherwise \end{cases} \tag{14}$$

Step Four: After updating the crow's position i its memory will also be updated as follows:

$$m^{i.it+1} = \begin{cases} x^{i,it+1}, & if\ \ F\left(x^{i,it+1}\right) < F\left(m^{i,it}\right) \\ m^{i,it}, & otherwise \end{cases} \tag{15}$$

where $F(.)$ represents the value of the objective function.

In CSA, all crows generate new positions and update their memory. These steps continue until the maximum number of iterations. Finally, the best memory response is selected as the CSA optimized response.

In CSA, abundance and diversity are mainly controlled by the awareness probability parameter (AP). By reduction of awareness probability value, CSA is tendentious to encourage search on a local area

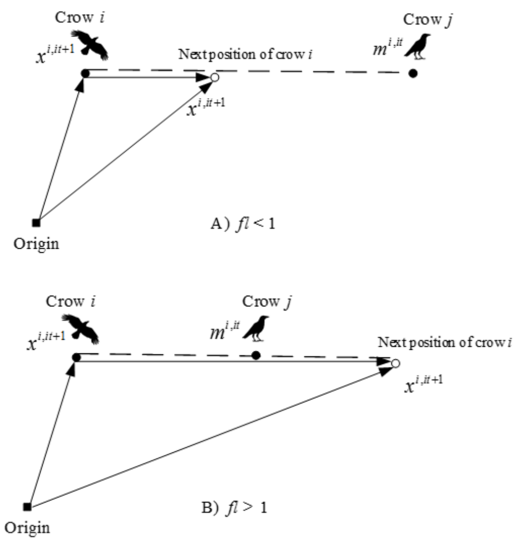

Figure 4. Schematic of the CSA mode for $fl < 1$ and $fl > 1$. Crow i can move along the line in different positions [30]

which is found by an appropriate response there. Thus, use of small values of AP, increases abundance. Furthermore, by increasing the AP value, the probability of searching for an area that is adjacent to the current appropriate response is less likely and CSA tends to have a global search space (randomly). Therefore, use of large values of AP increases abundance. Figure 5 displays the pseudo code for crow search algorithm [30].

3. CSAPF-SLAM ALGORITHM

As mentioned in the introduction, the SIR method for particle sampling is used in response to particle degradation problems. However, the SLAM problem still involves other issues such as sample depletion and high precision estimation with fewer particles. In response to these problems, modern particle filtering based on intelligent optimization mechanism of the biological group is a new development direction. The particles in the particle filter are considered as individual's in biological clusters and the distribution of particles is simulated using the motion laws of biological clusters. In other words, copying particles with higher weights and removing samples with lower weights and having the same history for the samples in the resampling process creates a challenge in resampling called particle impoverishment. In fact, the loss of particle diversity for estimation has led to the lack of particles. To overcome this problem, one solution is to diversify the selection of particles after resampling. That's why we go to biological group. In this article, the group of crows is considered as particles.

The following notes are considered in order to adapt the Crow Search Algorithm to the SLAM:

```
1: Randomly initialize the position of a flock of $N$ crows in the
   search space
2: Evaluate the position of the crows
3: Initialize the memory of each crow
4: while iter < $iter_{max}$
5:     for $i = 1{:}N$ (all N crows of the flock)
6:         Randomly choose one of the crows to follow (for
           example j)
7:         Define an awareness probability
8:             if $r_j \geq AP^{j,iter}$
9:             $x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} - x^{i,iter})$
10:            else
11:            $x^{i,iter+1} = a\ random\ position\ of\ search\ space$
12:            end if
13:    end for
14:    Check the feasibility of new positions
15:    Evaluate the new position of the crows
16:    Update the memory of crows
17:end while
```

Figure 5. pseudo code for crow search algorithm

1. Initial Population Creation: Initial population is assumed to be equal to FastSLAM particles and their relationships are as follows:

$$x_i = p_i + unifrnd\big(-\varepsilon, \varepsilon, size(p)\big) \tag{16}$$

where x and p are crow and particle position, respectively. ε is a small number, $size(p)$ represents particle dimension and $unifrnd(.)$ is a random uniform distribution. In fact, crows are produced using this method based on the particles and with a slight difference from particles to maintain Fast SLAM correlation.

2. Determine the fitness function of each crow: To have a criterion for updating a crow's memory, after updating its location, the appropriate fit function must also be selected.
3. Crow position updating: The following relation can be used to update the state of a crow:

$$x_k^i = \begin{cases} x_{k-1}^i + r_i \times fl_{k-1}^i \times \left(m_{k-1}^j - x_{k-1}^i\right) & rand \geq AP \\ p_k^i + unifrnd\big(-\varepsilon,\varepsilon,size(p)\big) & rand < AP \end{cases} \tag{17}$$

where m_j is the best crow position that crow i chases and $rand$ is an arbitrary random number.

Fast SLAM based on the crow search algorithm is obtained by substituting the position obtained from the crow search algorithm, i.e. Equation (17), in the observed position in Equation (8). Therefore, we will have the following for the mean value:

$$\underline{\tilde{x}}_k = \sum_{i=1}^{N_s} w_k^i \cdot x_{k,Crow}^i \tag{18}$$

where $x_{k,Crow}^i$ is obtained by the following recursive algorithm:

$$x_{k+1,Crow}^i = \begin{cases} x_{k,Crow}^i + r_i \times fl_k^i \times \left(m_k^j - x_k^i\right) & if\ r_j \geq AP_k^j \\ a\ random\ position & o.w \end{cases} \tag{19}$$

Figures 6 and 7 show the flow and pseudocode of proposed algorithm, respectively, that help us in implementation and simulation.

The fitness function in the proposed method is calculated based on an innovation. In fact, the calculated variance between distance of each particle from the average distance between each particle and position estimate at that moment is considered as a function of fitness. In other words:

$$F(\tilde{x}) = \sum_{i=1}^{N} \frac{x_i - x_{avg}}{x_N} \tag{20}$$

where F is the fitness function and $\tilde{x}$ is the estimated position using the proposed algorithm. The normalization factor (x_N) [8], and x_{avg} are calculated as follows:

$$x_N = \begin{cases} max\left|x_i - x_{avg}\right|_{1 \leq i \leq m}, & max\left|x_i - x_{avg}\right|_{1 \leq i \leq m} > 1 \\ 1 & , \quad others \end{cases} \tag{21}$$

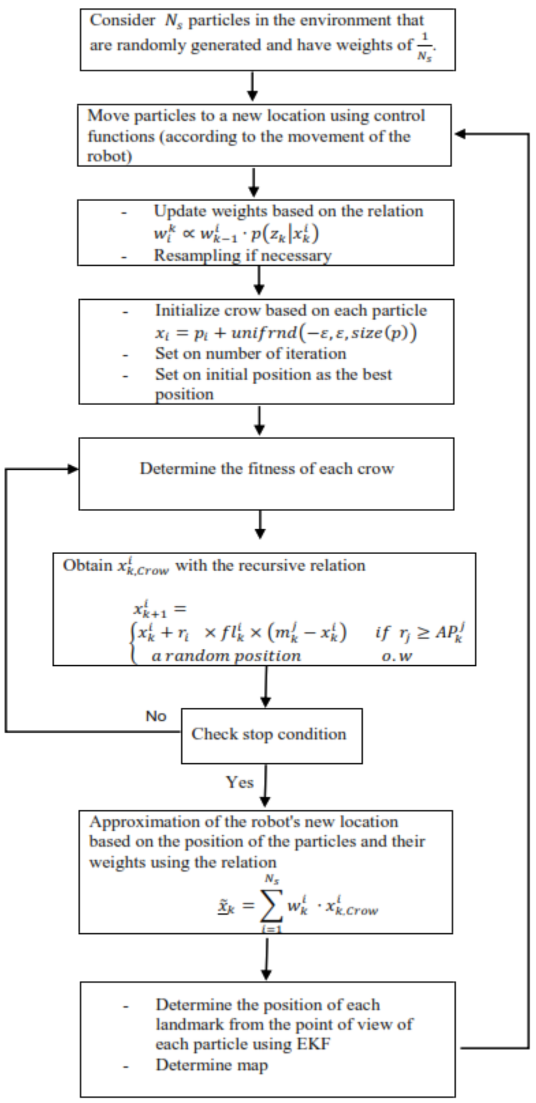
Consider N_s particles in the environment that are randomly generated and have weights of $\frac{1}{N_s}$.
Move particles to a new location using control functions (according to the movement of the robot)
- Update weights based on the relation $w_i^k \propto w_{k-1}^i \cdot p(z_k|x_k^i)$
- Resampling if necessary
- Initialize crow based on each particle $x_i = p_i + unifrnd(-\varepsilon, \varepsilon, size(p))$
- Set on number of iteration
- Set on initial position as the best position
Determine the fitness of each crow
Obtain $x_{k,Crow}^i$ with the recursive relation
$x_{k+1}^i = \begin{cases} x_k^i + r_i \times fl_k^i \times (m_k^j - x_k^i) & if\ r_j \geq AP_k^j \\ a\ random\ position & o.w \end{cases}$
No
Check stop condition
Yes
Approximation of the robot's new location based on the position of the particles and their weights using the relation
$\tilde{x}_k = \sum_{i=1}^{N_s} w_k^i \cdot x_{k,Crow}^i$
- Determine the position of each landmark from the point of view of each particle using EKF
- Determine map

Figure 6. Flow of the proposed algorithm

```
1: for $i = 1: N_s$
2:       randomly choose initial state $x_0^i$ in the search space
3:       $w_0^i = 1/N_s$
4: end for
5: k=1
6: while (k <> 0)
7: // New pose selection and update weights with new observations
8:    for $i = 1: N_s$
9:          sample a new pose $x_k^i \sim p(x_k|x_{k-1}^i, u_k)$
10:         $w_k^i = w_{k-1}^i \cdot p(z_k|x_k^i)$
11:      // Incorporate the measurement $z_k^i$ into the corresponding
                      EKF for mapping
12:         for $j = 1: m$
13:             update mean $\mu_{j,k}^i$
14:             update covariance $\Sigma_{j,k}^i$
15:         end for
16:    end for
17:    Apply the selected resampling method for the set of particles
      and their weights to get a new set of particles and weights
18:    // Initialize crow based on each particle
19:    for $j = 1: N_s$
20:       $x_i = p_i + unifrnd(-\varepsilon, \varepsilon, size(p))$
21:    end for
22:    Initialize the memory of each crow
23:    Set on number of iteration as $iter_{max}$
24:    while iter < $iter_{max}$
25:       for $i = 1: N_s$ (all N crows of the flock)
26:       Randomly choose one of the crows to follow (for example
              j)
27:             Define an awareness probability
28:             if $r_j = AP^{j,iter}$
29:                $x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times$
                                   $(m^{j,iter} - x^{i,iter})$
30:             else
31:                $x^{i,iter+1} =$
                        $a\ random position\ of\ search\ space$
32:             end if
33:       end for
34:       Check the feasibility of new positions
35:       Evaluate the new position of the crows
36:       Update the memory of crows
37:       Determine the fitness function of each crow
38:    end while
39:    // Calculate the estimate (Same as probability integral from
            probability density function)
40:    for $j = 1: N_s$
41:       $\tilde{x}_k = w_k^j . x_k^j$
42:       $x_k += \tilde{x}_k$
43:       Estimate map using $\mu_k^j$ and $\Sigma_k^j$
44:    end for
45:    $k = k + 1$
46:    if fitness function condition is realized
47:    $k = 0$
48:    end if
49:end while
```

Figure 7. Pseudo code for proposed algorithm

$$x_{avg} = \frac{1}{N} \sum_{i=1}^{N} |\tilde{x} - x_i| \tag{22}$$

4. EXPERIMENTAL EVALUATION

To evaluate the proposed method and compare it with the existing Fast SLAM method, we first deal with the problem solving scenario. In the scenario, we look at the problem-solving steps and the relationships between the sensor observations and the control functions to localization and estimate the position of the landmarks.

4. 1. Scenario

In general, our problem-solving scenario is as follows:

1. The robot starts moving from a point and spatial variations relative to that point are measured. At the starting point of motion, we take space containing N_s particles that are randomly generated and have weights of $1/N_s$.
2. The robot spends a moment along the way. In this case, we are faced with two issues:

A) Robot pose localization
It is clear that the particles must also be moved to new points. This particle transfer is applied to each particle by control functions (steering angle, speed, and noise).

$$\begin{bmatrix} x(k) \\ y(k) \\ \emptyset(k) \end{bmatrix} = f(x,u) = \begin{bmatrix} x(k-1) + \Delta T\left(v_c\cos(\phi) - \frac{v_c}{L}\tan(\phi)(a\sin(\phi) + b\cos(\phi))\right) \\ y(k-1) + \Delta T\left(v_c\sin(\phi) + \frac{v_c}{L}\tan(\phi)(a\cos(\phi) - b\sin(\phi))\right) \\ \phi(k-1) + \Delta T\frac{v_c}{L}\tan(\alpha) \end{bmatrix} \quad (23)$$

where v_c, L, a and b are robot velocity, the distance between the front and rear wheels, the distance between the laser and the center of the rear axle and the distance between the laser and the center of the front axle, respectively. In this way we moved the particles to a new position. Now we need to update the weights as well, which is based on the relation $w_i^k \propto w_{k-1}^i \cdot p(z_k|x_k^i)$. Obviously, sampling will also take place if necessary. After this step, particle impoverishment correction algorithms are performed by affecting location and weight. From the new position of the particles and their weights, the new location of the robot is approximated by Equation (8). If we use the CSAPF method, we conclude Equation (18) where $x_{k,Crow}^i$ is obtained by Equation (19).

B) Determining the location of landmarks
Based on the inverse of the observation model, which is the $l_{j,k}^i = h^{-1}(z_k,x_k^i)$, the position of each landmark is obtained from the point of view of each particle using EKF. In the following equation, $l_{j,k}^i$ represents the position of Landmark j from the perspective of particle i at time k, z_k is observation at time k, and x_k^i is the position of particle i at time k, which is obtained in section A. The function h, which is the observation model, has the following relations for the measurements of distance and angle sensors:

$$\begin{bmatrix} z_r \\ z_\theta \end{bmatrix} = h(x) = \begin{bmatrix} \sqrt{(x_l-x_v)^2+(y_l-y_v)^2} \\ \operatorname{atan}\left(\frac{(y_l-y_v)}{(x_l-x_v)}\right)-\phi+\frac{\pi}{2} \end{bmatrix} \quad (24)$$

The above equation shows that if the position of the robot is known in two-dimensional coordinates (x_v,y_v) and its orientation is ϕ and the landmark position is (x_l,y_l), h indicates the measurements received by the sensors. Obviously, by inverting this function, the location of the landmarks can be approximated based on the position of the robot and observation measurements.

3. The robot spends a moment on the path again. At this time, determining the position of the robot is like the second step. But landmark localization is divided into two categories:

A) Facing new landmarks: The position of the new landmarks that the robot encounters are determined as in step two.

B) Facing previous landmarks: If the robot sees the previous landmarks again, then the previous position is updated with new information. This happens with Kalman's gain. This way, using the h function, a prediction about the measurements is made from the previous position of the landmark and the current position of the robot. In other words $\hat{z} = h(l_{j,k-1}^i,x_k^i)$. Then, we get the updated position of this landmark using the $l_{j,k}^i = l_{j,k-1}^i + K(z_k,\hat{z})$ where K is Kalman gain. It should be noted that at any moment, we have assumed that the corresponding variables are known. This is one of the assumptions of using Fast SLAM.

After stating the scenario, to prove the claim, the simulation must be performed using valid datasets. In the next section, we have dealt with the issue.

4. 2. Simulation In order to test the proposed algorithm, we use simulator and presented dataset in Sydney University [26]. In this simulator, the robot is equipped with steering wheel angle sensors, laser and encoder. We used MATLAB code to simulate a particle filter with 150 particles. In this paper, we apply crow search algorithm to particle filter based SLAM. This paper compares localization accuracy of particle filter SLAM and CSAPF-SLAM.

1. System model

The schematic diagram of robot motion and robot observation is selected in accordance with what is shown in literature [12]

2. Performance analysis

In MATLAB platform, in a 100m × 100m area, original PF-SLAM and CSAPF-SLAM are simulated. Robot motion velocity, sensor observation range and AP factor are considered 0.5 m/s, 15m and 0.3, respectively [12]. System Noise and observation noise are considered 0.1. In this paper, the position of the landmarks is considered constant in all run. In fact, in order to be a good reference for comparison, the data received by the sensors on robot and consequently the position of the landmarks is considered constant. However, this does not mean that the robot or vehicle is aware of landmarks position and this does not diminish the universality of the method. Figures 8 and 9 demonstrate simulation results using Fast SLAM and the proposed CSAPF-SLAM algorithms respectively. Blue and red solid lines represent GPS information and estimated path of robot respectively. Green stars correspond to landmarks and red stars are their estimated position. Horizontal and vertical axis in both figures are in meters. Form Figures 8 and 9 it is found that CSAPF-SLAM algorithm has a better performance and is closer to the original path (GPS information) than PF-SLAM and estimated position of landmarks are more accurate. The mean square error (MSE) criterion has been used to confirm performance improvement. Figures 10 and 11 show the considerable improvement of the path and landmarks' position, respectively. The graphs show the effect of increasing the number of particles on the landmarks position estimation error and the path estimation error. As can be seen, the error rate in the proposed method has improved significantly compared to the Fast SLAM method, and in both methods the error rate has decreased with increasing number of particles. Vertical axis in both figures are in meters.

It should be noted that the MSE obtained for each particle is the result of the average number of times the program runs for that particle. In other words, due to the random nature of particle selection and the use of random distributions in parts of the crow search algorithm, similar to other meta-heuristic methods, with one run of simulation, proposed method performance improvement may not always be observed with increasing particle number. However, more program runs lead to the conclusion that the MSE mean decreases with increasing number of particles. The number of runs in our simulations was equal to 5.

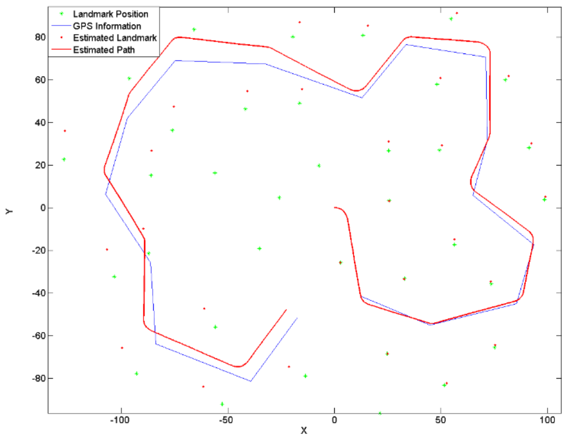

Figure 8. Bailey Simulator results for FastSLAM

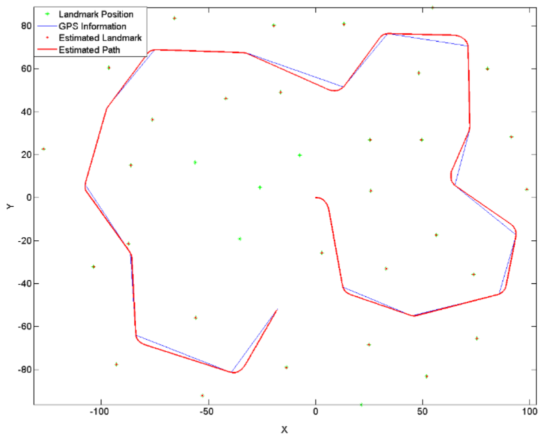

Figure 9. Bailey Simulator results for CSAPF-SLAM

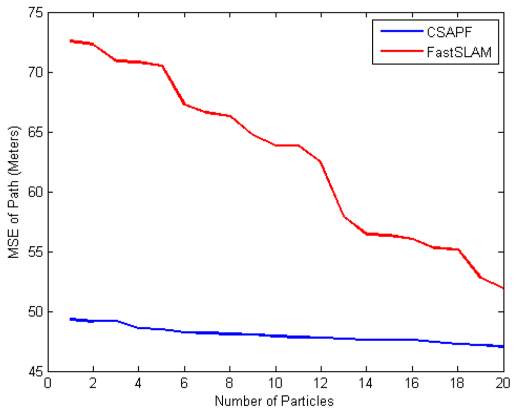

Figure 10. Mean Square Error between true and estimated path for Bailey Simulator

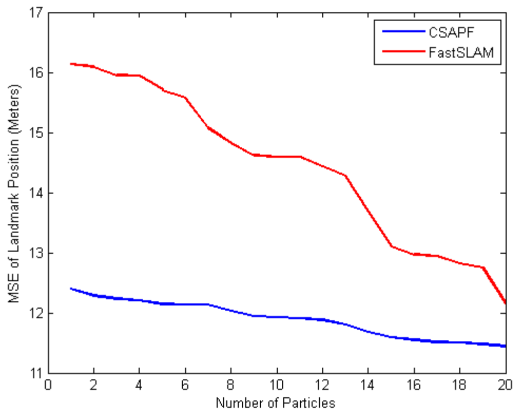

Figure 11. Mean Square Error between true and estimated landmarks position for Bailey Simulator

3. Verification using real-world dataset

To confirm the test in real conditions, a vehicle with the sensors mentioned in the previous section will travel outdoors for 30 minutes in the presence of trees for a distance of 4 km. So here, unlike the methods in which landmarks are moving [31], we consider them fixed. This data set is known as Victoria Park Dataset and we have used it in our work.

Figures 12 and 13 show the results of Fast SLAM and CSAPF-SLAM simulations for Victoria Park Dataset, respectively. The horizontal and vertical axes are in meters. In both figures, the continuous blue dots represent the GPS information and the red line represents the simulations from the two algorithms, and the stars represent the landmark position estimates.

As can be seen from the comparison of the two figures, the proposed CSAPF-SLAM algorithm is more consistent with the information obtained from GPS. Here too, the MSE criterion is used for comparison. Figure 14 shows the MSE for estimating the path in terms of the number of particles. In this figure, it is clear that the proposed CSAPF-SLAM algorithm achieves a significant improvement compared to Fast SLAM.

Another factor that is important to consider is the program running time. Although Figure 15 shows the longer time required to execute the proposed CSAPF method than Fast SLAM, but this amount of time increase versus increased accuracy seems negligible.

It can be seen that in the simulations, if we increase the number of particles, N_s in Figure 7, we get a higher accuracy estimate. Although by this we have lost more time for processing. Also, if we increase the velocity of the robot or vehicle, v_c in Equation (23), the

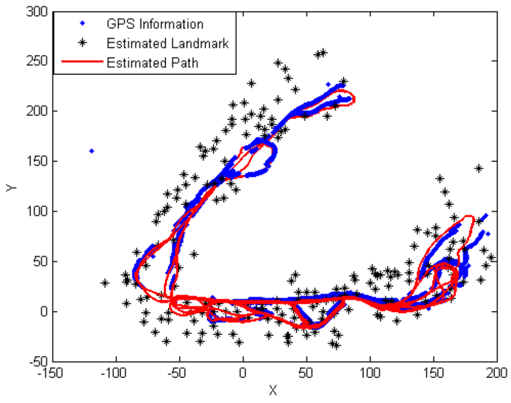

Figure 12. Fast SLAM simulation results using Victoria Park Dataset

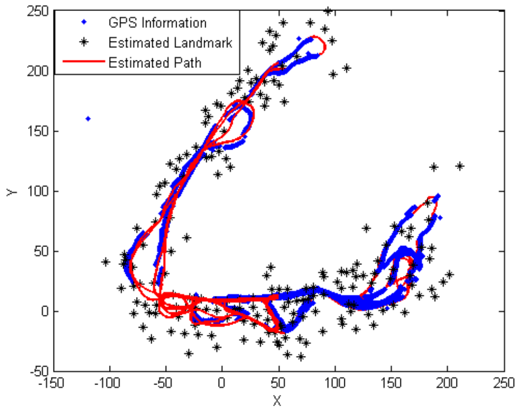

Figure 13. CSAPF-SLAM simulation results using Victoria Park Dataset

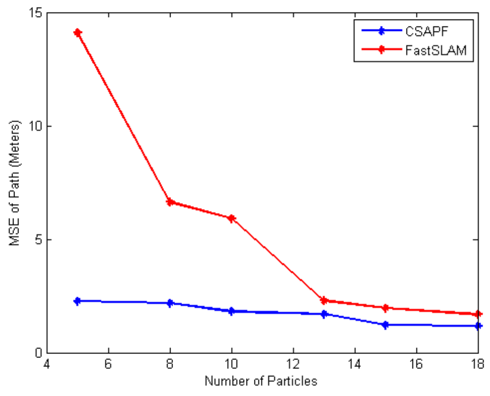

Figure 14. Mean Square Error between true and estimated path for Victoria Park Dataset

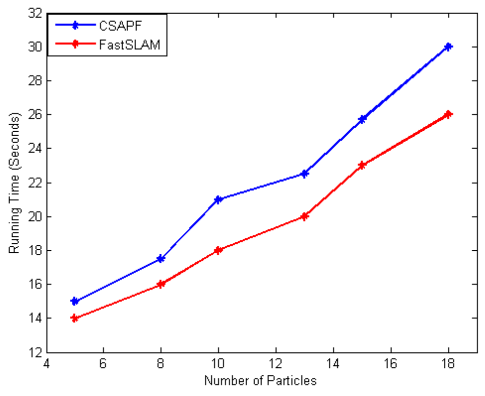

Figure 15. Running time of FastSLAM and CSAPF-SLAM for Victoria Park Dataset

observations and as a result the estimation will be less accurate. The next factor that can lead to a more accurate estimate is the use of a higher quality laser.

5. CONCLUSION

In all particle filter-based algorithms, using resampling to overcome particle degeneracy leads to particle impoverishment. This problem is also present in particle filter-based SLAM methods. Although the effect of particle impoverishment improvement is not directly observable, a reduction in particle impoverishment can be seen in improving the estimate. Therefore, our criterion for examining the reduction of particle impoverishment is the rate of improvement of the estimate.

There are two conventional ways to overcome this problem. The first is to diversify the choice of particle location and the other is to better allocate weight to the particles. In this paper, we select the first solution to overcome this problem and we present a new method called CSAPF-SLAM.

The proposed method was investigated using the Car Park Dataset and Victoria Park Dataset and the simulation results show that although more run time was spent, the estimation error was significantly reduced.

In the future, situations could be considered in which the environment would include moving objects in addition to stationary objects, or the problem of cooperative SLAM for multiple robots could be considered in which multiple robots and their paths could be identified.

Statements and Declarations

Funding

The authors declare that no funds, grants, or other assistance were received during the preparation of this manuscript.

Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

Author Contributions

All authors contributed to the conception and design of the study. Material preparation, data collection and analysis were performed by Hamid Jazayeriy, Javad Kazemitabar and Mohsen Abedini. The first draft of the manuscript was written by Mohsen Abedini, and all authors have commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Ethics Approval

This study uses the Victoria Park dataset. This dataset does not relate to humans or animals. Therefore, ethical approval is not applicable.

Consent to Participate

This paper does not include human participants who need to obtain consent.

Consent to Publish

This paper does not include human participants who need to obtain consent. The authors consent to the publication of this paper.

Data Availability

The data that support the findings of this study are openly available in Australian Centre for Field Robotics (ACFR) known as "Victoria Park Dataset" at http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm, [27].

6. REFERENCES

1. Smith, R., Self, M., and Cheeseman, P. "Estimating Uncertain Spatial Relationships in Robotics." In I. J. Cox & G. T. Wilfong (Eds.), ***Autonomous Robot Vehicles***, New York, NY: Springer New York, (1990), 167-193. https://doi.org/10.1007/978-1-4613-8997-2_14

2. Alitaleshi, A., Jazayeriy, H., and Kazemitabar, S. J. "WiFi Fingerprinting based Floor Detection with Hierarchical Extreme Learning Machine." In 2020 10th International Conference on Computer and Knowledge Engineering (ICCKE), 113-117. https://doi.org/10.1109/ICCKE50421.2020.9303624

3. Hooshmand, M., Yaghobi, H., and Jazaeri, M. "Irradiation and Temperature Estimation with a New Extended Kalman Particle Filter for Maximum Power Point Tracking in Photovoltaic Systems." ***International Journal of Engineering Transactions C: Aspects***, Vol. 36, No. 6, (2023), 1099-1113. https://doi.org/10.5829/ije.2023.36.06c.08

4. Yuen, D. C. K., and MacDonald, B. A. "An evaluation of the sequential Monte Carlo technique for simultaneous localisation and map-building." In Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Vol. 2, 1564-1569. https://doi.org/10.1109/robot.2003.1241817

5. Doucet, A., Godsill, S., and Andrieu, C. "On sequential Monte Carlo sampling methods for Bayesian filtering." ***Statistics and Computing***, (2000), 197-208. https://doi.org/10.1023/A:1008935410038

6. Talebi, Z., and Timarchi, S. "Improved distributed particle filter architecture with novel resampling algorithm for signal tracking." ***International Journal of Engineering Transactions C: Aspects***, Vol. 33, No. 12, (2020), 2482-2488. https://doi.org/10.5829/ije.2020.33.12c.07

7. Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem." In ***Proceedings of the 2002 AAAI National Conference on Artificial Intelligence,*** 593-598. https://doi.org/10.1.1.16.2153

8. Lei, X., Feng, B., Wang, G., Liu, W., and Yang, Y. "A novel fastSLAM framework based on 2D lidar for autonomous mobile robot." ***Electronics (Switzerland)***, Vol. 9, No. 4, (2020), 1-25. https://doi.org/10.3390/electronics9040695

9. Murphy, K. P. "Bayesian map learning in dynamic environments." ***MIT Press***, (2000), 1015-1021. https://doi.org/10.1.1.21.3240

10. Doucet, A., Freitas, N. de, Murphy, K. P., and Russell, S. J. "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks." In Proceedings of the 2000 16th Conference on Uncertainty in Artificial Intelligence, 176-183. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

11. Murphy, K., and Russell, S. "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks." In ***Sequential Monte Carlo Methods in Practice,*** New York, NY: Springer New York, (2001), 499-515. https://doi.org/10.1007/978-1-4757-3437-9_24

12. Zhu, D., Sun, X., Wang, L., Liu, B., and Ji, K. "Mobile robot SLAM algorithm based on improved firefly particle filter." Proceedings - 2019 International Conference on Robots and Intelligent System, ICRIS 2019, (2019), 35-38. https://doi.org/10.1109/ICRIS.2019.00018

13. Sadati Tilehboni, S. A., Jazayeriy, H., and Valinataj, M. "Genetic Algorithm with Intelligence Chaotic Algorithm and Heuristic Multi-Point Crossover for Graph Coloring Problem." ***Signal and Data Processing***, Vol. 14, No. 2, (2017), 75-95.

14. Sangdani, M. H., and Tavakolpour-Saleh, A. R. "Particle swarm optimization based parameter identification applied to a target tracker robot with flexible joint." ***International Journal of***

Engineering Transactions C: Aspects, Vol. 33, No. 9, (2020), 1797-1802. https://doi.org/10.5829/ije.2020.33.09c.14

15. Huang, C., Fei, J., Wang, L., and Liu, X. "Particle Filter Method Based on Multi-strategy Difference Cuckoo Search Algorithm." ***Nongye Jixie Xuebao/Transactions of the Chinese Society for Agricultural Machinery***, Vol. 49, (2018), 265-272. https://doi.org/10.6041/j.issn.1000-1298.2018.04.030
16. Chen, Z., Bo, Y., Tian, M., Wu, P., and Ling, X. "Dynamic Perceptive Bat Algorithm Used to Optimize Particle Filter for Tracking Multiple Targets." ***Journal of Aerospace Engineering***, Vol. 31, No. 3, (2018), 1-17. https://doi.org/10.1061/(ASCE)AS.1943-5525.0000834
17. Gao, M. L., Li, L. L., Sun, X. M., Yin, L. J., Li, H. T., and Luo, D. S. "Firefly algorithm (FA) based particle filter method for visual tracking." ***Optik***, Vol. 126, No. 18, (2015), 1705-1711. https://doi.org/10.1016/j.ijleo.2015.05.028
18. Tian, M. C., Bo, Y. M., Chen, Z. M., Wu, P. L., and Zhao, G. P. "Firefly algorithm intelligence optimized particle filter." ***Zidonghua Xuebao/Acta Automatica Sinica***, Vol. 42, No. 1, (2016), 89-97. https://doi.org/10.16383/j.aas.2016.c150221
19. Havangi, R. "A new modified particle filter with application in target tracking." ***Iranian Journal of Electrical and Electronic Engineering***, Vol. 16, No. 4, (2020), 449-460. https://doi.org/10.22068/IJEEE.16.4.449
20. Iswanto, Ma'arif, A., Raharja, N. M., Supangkat, G., Arofiati, F., Sekhar, R., and Rijalusalam, D. U. "Pid-based with odometry for trajectory tracking control on four-wheel omnidirectional COVID-19 aromatherapy robot." ***Emerging Science Journal***, Vol. 5, (2021), 157-181. https://doi.org/10.28991/ESJ-2021-SPER-13
21. Moghaddasi, S. S., and Faraji, N. "A hybrid algorithm based on particle filter and genetic algorithm for target tracking." ***Expert Systems with Applications***, Vol. 147, No. 7, (2020), 915-923. https://doi.org/10.1016/j.eswa.2020.113188
22. Shijing, D., Hongru, C., Xudong, W., Deshi, W., and Yongyong, Z. "Modal Optimization Design of Supporting Structure Based on the Improved Particle Swarm Algorithm." ***International Journal of Engineering, Transactions A: Basics***, Vol. 35, No. 4, (2022), 740-749. https://doi.org/10.5829/IJE.2022.35.04A.14
23. Rashno, A., and Fadaei, S. "Image Restoration by Projection onto Convex Sets with Particle Swarm Parameter Optimization." ***International Journal of Engineering Transactions C: Aspects***, Vol. 36, No. 2, (2023), 398-407. https://doi.org/10.5829/IJE.2023.36.02B.18
24. Samieiyan, B., MohammadiNasab, P., Mollaei, M. A., Hajizadeh, F., and Kangavari, M. "Novel optimized crow search algorithm for feature selection." ***Expert Systems with Applications***, Vol. 204, (2022), 117486. https://doi.org/https://doi.org/10.1016/j.eswa.2022.117486
25. Hussien, A. G., Amin, M., Wang, M., Liang, G., Alsanad, A., Gumaei, A., and Chen, H. "Crow search algorithm: Theory, recent advances, and applications." ***IEEE Access***, Vol. 8, , (2020), 173548-173565. https://doi.org/10.1109/ACCESS.2020.3024108
26. "Tim Bailey Simulator." Retrieved from http://www-personal.acfr.usyd.edu.au
27. Guivant, J. "Victoria Park Dataset." Retrieved from http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm
28. Hadian Jazi, S., Farahani, S., and Karimpour, H. "Map-merging in multi-robot simultaneous localization and mapping process using two heterogeneous ground robots." ***International Journal of Engineering, Transactions A: Basics***, Vol. 32, No. 4, (2019), 608-616. https://doi.org/10.5829/ije.2019.32.04a.20
29. Sanjeev Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." ***IEEE Transactions on Signal Processing***, Vol. 50, No. 2, (2002), 174-188. https://doi.org/10.1109/78.978374
30. Askarzadeh, A. "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm." ***Computers and Structures***, Vol. 169, (2016), 1-12. https://doi.org/10.1016/j.compstruc.2016.03.001
31. Badalkhani, S., and Havangi, R. "Effects of moving landmark's speed on multi-robot simultaneous localization and mapping in dynamic environments." ***Iranian Journal of Electrical and Electronic Engineering***, Vol. 17, No. 1, (2020), 1-10. https://doi.org/10.22068/IJEEE.17.1.1740

Persian Abstract

چکیده

ردیابی مسیر و موقعیت یابی در بسیاری از زمینه ها، از جمله ربات ها و وسایل نقلیه خودران ضروری است. در برخی موارد، مانند مناطقی که سیگنال های GPS ضعیف هستند یا در دسترس نیستند، ردیابی مسیر به عنوان یک سیستم موقعیت یابی جایگزین استفاده می شود. در این موارد، مکان‌یابی و نقشه‌برداری همزمان (SLAM) از اهمیت بالایی برخوردار است، زیرا نیازی به دانش قبلی و اثر انگشت آفلاین تجربی ندارد. SLAM را می توان با الگوریتم های پردازش سیگنال ترکیب کرد که در میان آنها فیلتر ذرات برجسته است. با این حال، چالش هایی مانند تباهی وزن ذرات و فقدان ذرات وجود دارد که باید با آنها مقابله کرد. در واقع، از دست رفتن تنوع ذرات برای تخمین منجر به فقدان ذرات می شود. برای غلبه بر این مشکل، یک راه حل، تنوع بخشیدن به انتخاب ذرات پس از نمونه برداری مجدد است. در این مقاله، ما یک الگوریتم جستجوی کلاغ (CSA)برای غلبه بر این مسائل و بهبود تخمین موقعیت پیشنهاد می‌کنیم. نتایج شبیه سازی نشان می دهد که این الگوریتم عملکرد FastSLAM را تا حد زیادی بهبود می بخشد.