A cubic spline method combing improved particle swarm optimization for robot path planning in dynamic uncertain environment

Wen Li¹, Mao Tan², Ling Wang³ and Qiuzhen Wang⁴

Abstract
This article considers a robot path planning problem originated from a robot factory inspection scenario. In the problem, the robot is in a dynamic uncertain environment, that is, a moving target object and several static and dynamic obstacles. An inertial positioning strategy is proposed to enable the robot to predict the position of the target in advance. From this predicted position, the robot path is generated by cubic spline interpolation, and then an improved particle swarm optimization algorithm with a random positive feedback factor in velocity updating optimizes the path. The experimental results show that the proposed method can successfully avoid the obstacles and reach the target object. In addition, the inertial positioning strategy and the improvement of particle swarm optimization can effectively shorten the path of the robot.

Keywords
Robot path planning, mobile robot, particle swarm optimization, cubic spline

Date received: 18 July 2019; accepted: 8 November 2019

Introduction
In recent years, with the rapid development of artificial intelligence (AI) technology, intelligent mobile robots have been widely used in many industrial fields, such as intelligent production, factory inspection, cargo handling, abnormal environment detection, underwater operations, and so on.¹,²,³ In the research of intelligent mobile robots, robot path planning, a key technology, has been a classical problem that has attracted the attention of many researchers. Generally, robot path planning problems can be classified in terms of two different evaluation metrics.⁴ One is based on environmental information, which separates the problems into static or dynamic path planning. In static path planning, environmental information and the position of the target do not change over time. In contrast, the position of the target or obstacle can change in dynamic path planning. The other one is based on the robot’s perception of the environmental information, which divides the problem into global and local path planning. Global path planning means that the robot can obtain global information, including information about all the obstacles, while local path planning is concerned with obtaining information about the immediate environment around the robot.
Path planning is known to be a non-deterministic polynomial (NP)-hard optimization problem. Many algorithms have been proposed to solve it. In the early stages of that research, some traditional and classical methods were used in robot path planning, for example, simulated annealing algorithm, artificial potential field (APF) method, fuzzy logic algorithm, tabu search algorithm, and the A* method.\(^{8,9,10,11,12}\) Another kind of method usually used is a graphical method, for example, viewable method, raster method, and the Voronoi diagram method.\(^{13,14,15}\) The mentioned traditional methods and graphical methods have some shortcomings, for example, the simulated annealing algorithm converges very slowly, the APF method finds it easy to fall into local optimality and the end point could be unreachable, for the A* algorithm and the Voronoi diagram method, the computing costs are very high. With the increase of the complexity of the environments and the difficulty of the task, the traditional methods usually find it difficult to achieve ideal performance.

In the past decades, a large number of studies and experiments have found that intelligent biological optimization algorithms have great advantages for robot path planning.\(^{5,16,17}\) For example, the ant colony algorithm, neural network algorithm, particle swarm algorithm, and genetic algorithm.

Yang et al.\(^{18}\) proposed a convolutional neural network to carry out path planning in a complex environment. However, that method is based on a grid environment: its effectiveness depends on the proximity of the real environment to a grid model. To simulate the environment more realistically, the mesh in the grid needs to be finer, but this will increase the computing cost. Hossain and Ferdous\(^{19}\) considered a problem with a lot of dynamic obstacles, and where the position of the target is unknown. Contreras-Cruz et al.\(^{20}\) designed a probabilistic roadmap method for these problems, but it cannot handle any unconsidered factors, which makes it impractical in real-world applications. Di Franco and Buttazzo\(^{21}\) proposed an energy-aware path planning algorithm to minimize energy consumption while satisfying a set of other requirements, for example, coverage and resolution. In traditional heuristic methods, it is always difficult to determine the optimal algorithm parameters. In this context, Karami and Hasanzadeh\(^{22}\) proposed an adaptive genetic algorithm to solve the problem, in which the parameters can be adjusted automatically. Davoodi et al.\(^{23}\) formulated two multi-objective path planning models to find a safe path, in which minimizing energy consumption is also considered. Besides the above-mentioned methods, Wang et al.\(^{24}\) proposed a deep reinforcement learning method to enhance the ability to explore unknown areas. A distributed multi-robot path planning algorithm based on a reserved area was proposed by Cao et al.\(^{25}\)

From the above analysis, we know that the existing research has mainly focused on minimizing the path length, smoothing the path, or minimizing the energy consumption. When the environment is dynamic and the acquisition of information about the environment is not timely or continuous, the path planning of mobile robots could become more complicated.

In this case, how to plan the shortest path as quickly as possible will play an important role in practice. This article considers such a robot path planning problem originated from a factory inspection scenario. In this problem, the environment consists of several static and dynamic obstacles, as well as a randomly moving target. We choose the cubic spline method to generate many possible paths and propose an improved particle swarm algorithm to obtain the optimal path.

The main contributions of this article are: (1) a new kind of path planning problem in a dynamic uncertain environment is considered; (2) an inertial positioning strategy is proposed to track the moving target, while the position of the target can only be received intermittently; and (3) an improved particle swarm optimization (PSO) is proposed and integrated with the cubic spline method to generate a shorter and smoother path.

The rest of this article is structured as follows. The second section describes the problem treated in this article. The third section focuses on the path planning method. In the fourth section, the results of computational experiments are provided so as to assess the effectiveness and performance of the proposed method. Finally, the conclusions of this article, as well as avenues for future research, are presented in the fifth section.

### Problem description

The so-called path planning problem refers to the process by which a robot plans a safe running path according to the sensors’ perception of the environment, and at the same time, it completes its task efficiently. The robot will follow the path to move step by step.

In this article, let us consider a scenario in intelligent factory as is shown in Figure 1. In a factory, a monitoring system integrated with a mobile robot is used to implement factory inspection. The robot has three key components to detect an object—a wireless receiver to receive the remote signal from monitoring system, an infrared detector to search surrounding objects, and a camera to identify near
objects. A path planning algorithm is running on the robot operating system. In the inspection procedure, if a suspicious object—maybe an unauthorized intruder, an animal, or other unknown mobile object—is detected by the camera, its position will be sent to the robot through the monitoring system. Due to the blind area of camera, the object position could be inconsistent and can be obtained intermittently. Utilizing the wireless signal, infrared signal, and camera signal, the robot can automatically plan a path to reach the suspicious object and avoid the obstacles, for example, the workshop, warehouse, and mobile forklift.

The given scenario in this work can be summarized as a path planning problem in a dynamic environment that consists of several static and dynamic obstacles, and the target object can be always moving. We assume the target is not moving randomly, but keeps its current motion state for a short time, so the trend of the target’s movement is predictable. The range of perception of the robot is limited, and the obstacles can be detected only when they are within this range. The robot can receive the signal of the position of the target. Due to limitations imposed by the environment, the robot can only acquire the position signal of the target intermittently, and the time interval of receiving the signal is uncertain. In this case, the objective of this problem is to find the shortest path from the current position of robot to the target object. Because the obstacles and targets are in motion, the robot’s path could also need to be updated with the changes of environment.

**Method**

In order to solve the problem, the robot’s path is generated by cubic spline interpolation from the predicted target position, and then an improved particle swarm optimization (IPSO) algorithm optimizes the path. Taking into account the fact that the environment is dynamically uncertain, an inertial positioning strategy is proposed to enable the robot to predict the position of the target in advance.

**Path generation by cubic spline interpolation**

Cubic spline interpolation is a piecewise method which forms a smooth line through a series of interpolation points, based on cubic polynomials. The robot’s path as fitted by cubic spline interpolation is smoother, which gives the robot better dynamic characteristics in an emergency stop or emergency steering, giving it great advantages over a path that consists of straight lines and arcs. Next, let us introduce the definition of cubic spline interpolation and the corresponding path generation method.

**Definition**

Given $n$ data points, the $i$th point is denoted by $(x_i, y_i)$, no two points are the same and $a = x_0 < x_1 < \cdots < x_n = b$. The spline $S(x)$ is a function satisfying:

1. $S(x) \in C^2[a, b]$;
2. On each subinterval $[x_{i-1}, x_i]$, $S(x)$ is a polynomial of degree 3, where $i = 1, \cdots, n$;
3. $S(x_i) = y_i$, for any $i = 0, 1, \cdots, n$.

The method of generating a path by cubic spline interpolation consists, to begin with, in the selection of $M$ interpolation nodes $(x_1', y_1'), (x_2', y_2'), \ldots, (x_M', y_M')$ between the starting point $(x_0, y_0)$ and the target point $(x_{n+1}, y_{n+1})$, which determine intervals $[x_0, x_{m_1}], \ldots, x_{m_n}$ and $[y_0, y_{m_1}], \ldots, y_{m_n}$. The cubic spline nodes are used to obtain $n$ interpolation points $(x_1, \ldots, x_n)$ and

![Figure 1. A scenario of factory inspection using a mobile robot.](image-url)
n interpolation points \((y_1,\ldots,y_n)\), and the points \((x_0,y_0),(x_1,y_1),\ldots,(x_{n+1},y_{n+1})\) are connected to form a continuous path. The path calculated by the robot according to the current environment is

\[
(x_0,y_0) \rightarrow (x_1,y_1) \rightarrow \ldots \rightarrow (x_n,y_n) \rightarrow (x_{n+1},y_{n+1})
\]

In order to find the shortest collision-free path, we can define the function

\[
\text{Fit} = L \cdot (1 + \gamma \cdot \text{len}/L)
\]

where

\[
L = \sum_{i=0}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}
\]

\[
\text{Len} = \sum_{k=1}^{n} \text{len}_k
\]

and \(\text{len}_k\) is the distance that the path goes through in the \(k\)th obstacle. \(\gamma\) is the coefficient to avoid obstacles, which is set to 200 in this article. \(L\) in equation (1) is used to find the shortest path, and the part in brackets is used to make the robot avoid obstacles.

**An improved PSO for path optimization**

PSO is a swarm intelligence optimization algorithm, which was first proposed by Kennedy and Eberhart (1995). The PSO algorithm was derived from research on the predatory behavior of birds. When birds hunt for food, the simplest and most effective strategy is to search the area surrounding the bird which is closest to the food. In order to achieve the goal of approaching the location of the food, each bird learns from its personal best location (pbest) it has experienced and the global best location (gbest) in the population to finally approach the location of the food. The mathematical description of the PSO algorithm is as follows. In a \(d\)-dimensional search space, there is a population \(X\) consisting of \(N\) particles \(X = (X_1,X_2,\ldots,X_N)\), where particle \(i\) is described by a \(d\)-dimensional vector \(X_i = (x_{i1},x_{i2},\ldots,x_{id})\), which represents the position of particle \(i\) in the \(d\)-dimensional search space, actually, \(X_i\) also means a potential solution of the problem. According to the objective function, a fitness value corresponding to each particle’s position \(X_i\) can be calculated. In this article, the fitness function can be defined as in equation (1). The velocity of particle \(i\) is \(V_i = (v_{i1},v_{i2},\ldots,v_{id})\), the individual extreme value is \(P_{i\text{best}} = (p_{i1\text{best}},p_{i2\text{best}},\ldots,p_{id\text{best}})\), and the population extreme value is \(G_{\text{best}} = (g_{1\text{best}},g_{2\text{best}},\ldots,g_{d\text{best}})\). The formulas for updating the particle’s velocity and position are

\[
V_i = w \cdot V_i + c_1 \cdot r_1 \cdot (P_{i\text{best}} - X_i) + c_2 \cdot r_2 (G_{\text{best}} - X_i)
\]

\[
X_i = X_i + V_i
\]

where \(w\) is the inertia weight, \(i = 1,2,\ldots,N\), \(i\) is the particle number, \(v_{id}\) is the velocity of the particle, \(c_1\) and \(c_2\) are the individual and social acceleration factors, and \(r_1\) and \(r_2\) are random numbers in \((0,1)\).

As we all know that in swarm intelligence algorithm, there is always a trade-off between the exploration ability and the exploitation ability. The purpose of PSO improvement is to enhance the diversity of population and prevent premature convergence, this improvement is expected to improve the exploration ability or the exploitation ability during different stages of the algorithm running, and also it is helpful to make the algorithm more stable in different environments. Employing stratified random perturbation, the diversity of the population is improved and the search space of particles is expanded in the improved algorithm.

In the IPSO, a random positive feedback factor is added into the velocity updating formula of the particles, which enhances the ability to find a local optimum. The hierarchical random initialization strategy is adopted to improve the global search ability of the population. Finally, the path planning of the robot is carried out in a simple and complex dynamic uncertain environment.

In order to enhance the exploitation ability, a random positive feedback factor \(\bar{E}\) is added into the velocity updating formula of the algorithm. The improved velocity updating rule then becomes

\[
V_i = w \cdot V_i + c_1 \cdot r_1 \cdot (P_{i\text{best}} - X_i) + c_2 \cdot r_2 (G_{\text{best}} - X_i) + \theta_i
\]

and the updating rule for \(\theta_i\) is changed correspondingly.

If \(\text{Fit}_i^k < \text{Fit}_i^{k-1}\)

\[
\theta_i = 1/k \cdot \theta_i + (X_i^k - X_i^{k-1}) \cdot \text{rand}(0,1)
\]

otherwise

\[
\theta_i = 1/k \cdot \theta_i - (X_i^k - X_i^{k-1}) \cdot \text{rand}(0,1)
\]

where \(\text{Fit}_i^k\) is the fitness value of \(i\)th particle in the \(k\)th iteration, and \(\text{rand}(0,1)\) generates a random value in \((0,1)\). More details of the particle updating process to improve the PSO are provided in Algorithm 1.

For further understanding of the proposed cubic spline method combing PSO (PSO-Sp), a diagram of path optimization by the improved PSO algorithm is shown in Figure 2.

**Inertial positioning method**

The robot can receive the position signal of the target, which may be an active signal, for example, radar signal, or a passive signal, for example, distress signal. Due to environmental limitations, the robot can only acquire the position signal of the target intermittently. In this case, the task of this problem is to find the shortest path between the starting point and the target point for the robot, and the path can be updated with any change of the environment.
Algorithm 1. Particle updating process of IPSO.

**Input:** \((x_0, y_0), (x_{n+1}, y_{n+1})\)

**Initialization:** Set parameters \(\text{IterNum}, N, c_1, c_2, w, \theta_i = 0\)

Initialize particles that randomly distributed between the starting and ending points.
Calculation the initial fitness value \(F_{\text{value}}\) according to Eq. (1).
Evaluate \(G_{\text{best}}\).

**Process:**
1: for \(k = 1 : \text{IterNum}\)
2: for \(i = 1 : N\)
3: \hspace{1em} Calculate particle velocity \(V_i\) according to Eq. (6).
4: \hspace{1em} Update particle position \(X_i\) according to Eq. (5).
5: \hspace{1em} Update \(\theta_i\) according to Eq. (7) – Eq. (8)
6: end
7: Sort particles by fitness value in ascending order.
8: for \(i = 1 : N \times 0.3\)
9: \hspace{1em} if \(\text{rand}(0,1) < 0.1\)
10: \hspace{2em} \(V_i = V_i + \text{rand}(0,1)\).
11: \hspace{2em} \(X_i = V_i + X_i\).
12: \hspace{1em} end
13: end
14: for \(i = N \times 0.3 + 1 : N\)
15: \hspace{1em} if \(\text{rand}(0,1) < 0.1\)
16: \hspace{2em} Re-initialize Particle \(i\).
17: \hspace{1em} end
18: end
19: Update \(P_{\text{best}}, G_{\text{best}}\).
20: end

**Output:** \(G_{\text{best}}\)

---

Figure 2. A diagram of path optimization.
In this article, we assume that the target is always moving during the search procedure, and the robot can only receive the position signal of the target intermittently. These assumptions are very natural and realistic, for example, in remote search and rescue, the communication is interrupted in some extreme environments. This will greatly affect the generation of the optimal path, and then delay the rescue. Therefore, this article proposes an inertial positioning method to quickly track the moving target. Inertial positioning is inspired by the law of inertia. Without external forces, any object will keep moving in a straight line. In this article, we make two assumptions on the motion of the target. One is that the target has the property of keeping its current state of motion. This means that the target is not moving randomly: the state of motion will not change or change very little in a short time. The other is that the movement of the target has a certain rhythm at the macro level, which means we can roughly judge the trend of the movement of the target. Therefore, in order to effectively predict the position of the target with limited information, the velocity prediction model of the target should meet the following conditions: the model must reflect the impact of the historical velocity on the current velocity. The closer the current time, the greater the impact. In this context, we propose an inertial positioning method to track the moving target.

Here, \( V^i(X, Y) \) is the predicted velocity of the target, \( T^i(x_i, y_i) \) is the detected position of the target at time \( i \), \( S(x_0, y_0) \) is the starting position of the robot, and \( P^i(x_i, y_i) \) is the predicted position of the target, \( \Delta V^i \) is the trend of change in the speed, which contains the predicted deviation and velocity deflection, and \( \Delta t_i \) is the interval between the \( i - 1 \)th and \( i \)th acquisitions of the position of the target. Then, the procedure for calculating \( P^i(x_i, y_i) \) is described in Algorithm 2. Figure 3 is a simple schematic diagram of the calculation of the predicted velocity \( V^i(X, Y) \). \( V^i(X, Y) \), \( T^i(x_i, y_i) \), \( P^i(x_i, y_i) \), and \( S(x_0, y_0) \) in the figure are the same as mentioned before. The solid line denotes the trajectory of the target. The vectors \( A \), \( B \), and \( C \) in Figure 3 correspond to \( A \), \( B \), and \( C \) in Algorithm 2. The arrows with the same color represent the same vector, in Figure 3, they are translated to make the calculation easier to understand. In addition, a simple flow chart of path planning is shown in Figure 4, from the figure we can see that the robot will move step by step and invoke our inertial positioning method and IPSO-Sp to update its path.

**Experimental results and analyses**

In this article, the path planning in dynamic and static environment is both evaluated and analyzed. In a dynamic environment, we mainly focus on the difference between paths with and without introducing the inertial positioning. In the static environment, we compared our improved PSO with standard PSO, genetic algorithm (GA), ant colony optimization (ACO) algorithm, differential evolution (DE) algorithm, fuzzy logic method, APF method, and A* to evaluate the performance improvement of our algorithm.

The IPSO-Sp, PSO-Sp, GA-Sp, DE-Sp, ACO, fuzzy logic, APF, and A* algorithms in this article are implemented on
the same software and hardware platform, the programming environment is MATLAB R2018a. In the experiment, there are two interpolation nodes and the number of interpolation points is 100. The key parameters for the PSO-Sp and IPSO-Sp algorithm are both set as follows: population size \( N = 20 \), maximum number of iterations \( \text{IterNum} = 100 \), inertia weight \( w = w \times 0.99 \), and initial \( w = 1 \). The learning factors \( c_1 = 2, c_2 = 2 \), \( V_{\text{max}} = 10 \). In order to guarantee the fairness of the algorithm comparison, the population size and number of iterations of GA-Sp and DE-Sp are set as the same as those of PSO-Sp. The unique parameters of each algorithm are determined according to the standard algorithm or empirical evidence.

**Effectiveness of inertial positioning strategy**

Two different dynamic environments were used for the path planning experiment to test the effectiveness of the inertial positioning method. Environment 1 consists of two dynamic obstacles, four static barriers, and a moving target object. Environment 2 includes only 10 static obstacles and 1 moving target. The robot path with and without inertial positioning in environments 1 and 2 is shown in Figures 5 to 8.

As shown in Figures 5 and 6, the dynamically generated paths without or with inertial positioning both make the robot arrive at the target position after several limited steps, and the obstacles are avoided in the path, which indicates that our proposed method is effective at implementing robot path planning in the given environment. However, we can see that the robot path with inertial positioning is significantly shorter than that without inertial positioning, so the robot can arrive at the target in fewer steps. The main reason for this is that the inertial positioning simulates the intuitive motion law of most objects: keep the speed and direction of motion roughly unchanged for a certain period of time; in most cases, the change of velocity is continuous, rather than jumping. These results show that inertial positioning could be useful in some practical cases, for example, when the communication is interrupted in some extreme environments.

The results in environment 2 are also shown in Figures 7 and 8, from which similar phenomena can be observed and the same conclusions can be drawn. Due to the randomness of the PSO algorithm, the computational results of our proposed method may be more or less different in each run. In order to evaluate our method more adequately and reliably, we executed the method in the same environment 100 times. The average path length and moving steps of the robot are listed in Table 1. From the table, we can see that the performance of the algorithm with inertial positioning is clearly improved in both environments, which is

---

**Figure 3. Prediction of the position of the target.**

**Figure 4. Flow chart of path planning.**
Figure 5. Robot path without inertial positioning in environment 1: (a) initial environment 1, (b) after 17 steps, (c) after 50 steps, and (d) after 76 steps.

Figure 6. Robot path with inertial positioning in environment 1: (a) initial environment 1, (b) after 20 steps, (c) after 32 steps, and (d) after 65 steps.

Figure 7. Robot path without inertial positioning in environment 2: (a) initial environment 2, (b) after 34 steps, (c) after 71 steps, and (d) after 81 steps.

Figure 8. Robot path with inertial positioning in environment 2: (a) initial environment 2, (b) after 34 steps, (c) after 46 steps, and (d) after 68 steps.
convincing evidence that the prediction-based positioning is effective and stable in the path planning.

**Performance analysis of the IPSO algorithm**

In this article, we proposed an improved PSO to optimize the planning of a path for a robot. In each iteration of the dynamic planning procedure, the optimization of the path planning is actually a static problem. To further evaluate the performance of the algorithm, two different static environments have been used to conduct comparative experiments. In both environments, there are five obstacles at different locations, and the target object is static. We conducted the computational experiment 100 times in static environments 3 and 4, A* algorithm, APF, fuzzy logic, ACO, and the cubic spline method integrated with DE, GA, standard PSO, and our improved PSO, respectively, are employed to provide comparative analysis in the experiments. These methods are referred as A*, APF, fuzzy, ACO, DE-Sp, GA-Sp, PSO-Sp, and IPSO-Sp in this article. The optimal feasible robot paths in the two environments are represented by different colors as shown in Figure 9. The paths that have the same way to avoid obstacles but longer than the listed optimal paths are not shown in the figure. We can see that the planned paths obtained by GA-Sp, PSO-Sp, and IPSO-Sp have the similar way to avoid obstacles.

Figure 10 presents the lengths of the paths obtained by different methods during the 100 experiments. From the figure, we can see that the path obtained by DE-Sp, GA-Sp, PSO-Sp, and IPSO-Sp is better than that of A*, APF, ACO, and fuzzy. Furthermore, we can see that most of the time, IPSO-Sp can obtain a significantly shorter path than PSO-Sp and GA-Sp, and the worst path generated by IPSO-Sp is still better than that generated by PSO-Sp and GA-Sp. Table 2 presents the statistical data of Figure 10 and shows that the paths found by IPSO-Sp are shorter in most cases than those from the other three algorithms in terms of the longest length, shortest length, and average length. The results indicate that our improvement of the PSO is effective at achieving higher performance for the proposed method.

In environment 3, only IPSO-Sp is relatively stable at finding paths, and in environment 4, both PSO-Sp and IPSO-Sp are stable, but our proposed IPSO-Sp outperforms PSO-Sp in finding a shorter path. The stability of the algorithms is affected by the environment features significantly. As is shown in Figure 9, there is a nearly symmetric path for each path in environment 4, in environments like this, the lengths for these paths are nearly the same. But in environment 3, different ways to avoid obstacles may lead to paths with different length, then some unstable algorithms could obtain not good paths.

Moreover, the convergence of DE-Sp, GA-Sp, PSO-Sp, and IPSO-Sp is also analyzed. Figure 11 depicts the fitness

![Figure 9. Optimal feasible paths: (a) environment 3 and (b) environment 4.](image-url)
Figure 10. Comparison of path length: (a) environment 3 and (b) environment 4.

Table 2. Statistical results and comparison in static environment.

| Environment | Algorithm | A*    | APF   | Fuzzy | ACO   | DE-Sp | GA-Sp | PSO-Sp | IPSO-Sp |
|-------------|-----------|-------|-------|-------|-------|-------|-------|--------|---------|
| 3           | Longest length | 103.19 | 107.29 | 117.89 | 107.02 | 104.94 | 102.81 | 99.47  | 99.25   |
|             | Shortest length | 103.19 | 107.29 | 117.89 | 102.88 | 98.82  | 99.64  | 98.59  | 98.04   |
|             | Average length | 103.19 | 107.29 | 117.89 | 102.98 | 99.38  | 100.42 | 98.91  | 98.48   |
| 4           | Longest length | 102.46 | 110.58 | 112.74 | 112.78 | 106.26 | 104.83 | 102.63 | 98.69   |
|             | Shortest length | 102.46 | 110.58 | 112.74 | 102.27 | 97.24  | 99.23  | 98.20  | 97.15   |
|             | Average length | 102.46 | 110.58 | 112.74 | 109.23 | 100.67 | 100.17 | 99.17  | 98.12   |

APF: artificial potential field; ACO: ant colony optimization; DE: differential evolution; PSO: particle swarm optimization; IPSO: improved particle swarm optimization.

Figure 11. Comparison of fitness value: (a) environment 3 and (b) environment 4.
Conclusion

In this article, an IPSO-Sp is proposed to solve the robot path planning problem in a dynamic uncertain environment. Experimental results show that the proposed inertial positioning strategy can significantly shorten the path in a dynamic environment and speed up the search process. Compared with the classical A*, APF, ACO, and fuzzy methods, the DE-Sp, GA-Sp, and PSO-Sp are more efficient in the dynamic uncertain environment. Compared with the GA-Sp and PSO-Sp, the proposed IPSO-Sp has better performance in finding the shorter path, stably, which was our main purpose for improving the standard PSO. Future research could be further improving the performance of the path planning method in more complicated environments, for example, irregular obstacles.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the Natural Science Foundation of China (61873222).

ORCID iD

Mao Tan https://orcid.org/0000-0002-2246-440X

References

1. He B, Wang S and Liu YJ. Underactuated robotics: a review. Int J Adv Robot Syst 2019; 16(4): 1–29.
2. Zhang LJ, Hu RQ and Yi WM. Research on force sensing for the end-load of industrial robot based on a 6-axis force/torque sensor. Acta Automatica Sinica 2017; 43(3): 439–447.
3. He B, Shao YW, Wang S, et al. Product environmental footprints assessment for product life cycle. J Clean Product 2019; 233: 446–460.
4. Goel AK and Callantine TJ. An experience-based approach to navigational route planning. In: Proceedings of the IEEE/RSJ international conference on intelligent robots and systems, Raleigh, NC, USA, 7–10 July 1992, Vol. 2, pp. 705–710. IEEE.
5. Mac TT, Copot C, Tran DT, et al. Heuristic approaches in robot path planning: a survey. Robot Autonom Syst 2016; 86: 13–28.
6. Das PK, Behera HS and Panigrahi BK. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm Evolution Computat 2016; 28: 14–28.
7. Zhu D and Yan M. Survey on technology of mobile robot path planning. Cont Dec 2010; 25(7): 961–967.
8. Turker T, SahingoZ OK and Yilmaz G. 2D path planning for UAVs in radar threatening environment using simulated annealing algorithm. In: 2015 International conference on unmanned aircraft systems (ICUAS), Denver, CO, USA, 9–12 June 2015, pp. 56–61. IEEE.
9. Montiel O, Orozco-Rosas U and Seplveda R. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. Exp Syst Appl 2015; 42(12): 5177–5191.
10. Bakdi A, Bentout A, Boutami H, et al. Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. Robot Autonom Syst 2017; 89: 95–109.
11. Panda MR, Priyadarshini R and Pradhan SK. Autonomous mobile robot path planning using hybridization of particle swarm optimization and tabu search. In: 2016 IEEE international conference on computational intelligence and computing research (ICCIC), Chennai, India, 15–17 December 2016, pp. 1–7. IEEE.
12. Zhang JW, Liu L and Chen K. Omni-directional bipedal walking path planning. Acta Automat Sinica 2016; 42: 189–201.
13. Toan TQ, Sorokin AA and Trang VTH. Using modification of visibility-graph in solving the problem of finding shortest path for robot. In: 2017 International Siberian conference on control and communications (SIBCON), Astana, Kazakhstan, 29–30 June 2017, pp. 1–6. IEEE.
14. Boschian V and Pruski A. Grid modeling of robot cells: a memory-efficient approach. J Intell Robot Syst 1993; 8(2): 201–223.
15. Lau B, Sprunk C and Burgard W. Efficient grid-based spatial representations for robot navigation in dynamic environments. Robot Autonom Syst 2013; 61(10): 1116–1130.
16. Cekmez U, Ozsiginan M and SahingoZ OK. Multi colony ant optimization for UAV path planning with obstacle avoidance. In: 2016 International conference on unmanned aircraft systems (ICUAS), Arlington, VA, USA, 7–10 June 2016, pp. 47–52. IEEE.
17. Hwu T, Wang AY, Oros N, et al. Adaptive robot path planning using a spiking neuron algorithm with axonal delays. IEEE Trans Cognit Develop Syst 2017; 10(2): 126–137.
18. Yang G, Deng Y, Zhuang X, et al. An improved real-time path planning of mobile robot in a complex and dynamic environment. In: First international conference on electronics instrumentation and information systems (EIIS), Harbin, China, 3–5 June 2017, pp. 1–4. IEEE.
19. Hossain MA and Ferdous I. Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. Robot Autonom Syst 2015; 64: 137–141.
20. Contreras-Cruz MA, Ayala-Ramirez V and Hernandez-Belmonte UH. Mobile robot path planning using artificial
bee colony and evolutionary programming. *Appl Soft Comput* 2015; 30: 319–328.

21. Di Franco C and Buttazzo G. Energy-aware coverage path planning of UAVs. In: *2015 IEEE international conference on autonomous robot systems and competitions*, Vila Real, Portugal, 8–10 April 2015, pp. 111–117. IEEE.

22. Karami AH and Hasanzadeh M. An adaptive genetic algorithm for robot motion planning in 2D complex environments. *Comput Electric Eng* 2015; 43: 317–329.

23. Davoodi M, Panahi F, Mohades A, et al. Clear and smooth path planning. *Appl Soft Comput* 2015; 32: 568–579.

24. Wang K, Bu XI, Li RF, et al. Path planning for robots based on deep reinforcement learning by depth constraint. *J Huazhong Univer Sci Technol (Nat Sci Ed)* 2018; 46(12): 77–82.

25. Cao QX, Huang XQ, Zhu XX, et al. Distributed multi-robot path planning based on reserved area. *J Huazhong Univer Sci Technol (Nat Sci Ed)* 2018; 46(12): 71–76.

26. Fritsch FN and Carlson RE. Monotone piecewise cubic interpolation. *SIAM J Numer Anal* 1980; 17(2): 238–246.

27. Kennedy J and Eberhart R. Particle swarm optimization (PSO). In: *Proceedings IEEE international conference on neural networks*, Perth, WA, Australia, 27 November–1 December 1995, pp. 1942–1948. IEEE.