A Path Planning Algorithm Based on A Heuristic Method

Xiao-Zhen Yan[1], Xin-Yue Zhou[1], Ruo-Chen Ding[1], Qing-Hua Luo[1,2], Chun-Yu Ju[1]*

[1] School of Information Science and Engineering, Harbin Institute of Technology at WeiHai, China
[2] Shandong Institute of Shipbuilding Technology, China
yxz_heu@163.com, zxyuelu@163.com, 1015412921@qq.com, luoqinghua081519@163.com, 2410242150@qq.com

Abstract

The path planning algorithm plays a pivotal role in the field of intelligent robotics. However, in certain scenarios, the A-star algorithm exhibits drawbacks such as excessive redundant nodes and prolonged path lengths. Consequently, this paper introduces a path planning algorithm employing a heuristic approach, which will be briefly elucidated in the subsequent sections. If the straight-line path from the starting point to the destination is unobstructed, iterative calculations for path planning are unnecessary. Should the straight-line path intersect obstacles, the proposed heuristic algorithm is employed for local path planning to circumvent obstacles, and the obtained path is subsequently integrated with the overall trajectory. In the same context, we compare this algorithm with other path planning algorithms, revealing that the enhanced algorithm presented in this paper achieves a reduction in path length ranging from 0.07% to 12.58%. This successfully validates the effectiveness of the improved algorithm proposed in this study.

Keywords: A-star algorithm, Path planning algorithm, Adaptive obstacle size, Heuristic method

1 Introduction

In the field of robotics, the study of path planning algorithms is of great significance. Intelligent unmanned equipment, such as unmanned aerial vehicle [1], mobile robots, self-driving cars [2] and unmanned surfaced vehicles [3], cannot achieve satisfactory navigation effects without path planning algorithms. The mobile robot uses its sensors to map the surrounding environment, a satisfactory path is produced by a path planning algorithm on a known map, and then transfer the results planned by the planner give the control layer to realize autonomous navigation of the robot. Overall, the application of path planning technology in intelligent robots has received much attention in recent years.

The solution to this problem can promote the application of robots in various scenes. For a mobile robot, to reach the designated position smoothly, it is necessary to find a path that is free of obstacles, short in length and with as few turning points as possible [4]. Domestic and international experts have conducted many studies on this issue. Generally, path planning algorithms can be divided into two main categories. We mainly give a detailed introduction to the global path planning algorithm [5].

The global path planning algorithms include the traditional algorithms and the intelligent algorithms.

In the realm of conventional algorithms, the Dijkstra algorithm, invented by Dutch computer scientist Edsger Wybe Dijkstra, calculates the shortest distance search by accumulating path lengths [6]. The algorithm involves continually inspecting all nodes adjacent to the current node in the set of nodes that have not yet found the shortest path [7]. Due to its non-directional search approach, the efficiency of the Dijkstra algorithm is notably low. Consequently, in 1968, Nils John Nilsson developed an enhancement to the Dijkstra algorithm, known as the A-star algorithm, which evaluates paths through a heuristic function to reduce the search scope and computational complexity [8-10]. Anthony Stenz extended the A-star algorithm in 1994, naming it the D-star algorithm, applicable to path planning in partially or completely unknown dynamic environments [11]. However, the D-star algorithm plans the next step within a finite time, and each step taken may not necessarily be the optimal path [12]. Rapidly Exploring Random Trees (RRT) refer to an algorithm forming a randomly expanding tree, with the starting point as the root node, adding leaf nodes through random sampling when overlapping with the endpoint occurs. The sampling method employed here is the incremental sampling search method [13]. While RRT is suitable for path planning problems in high-dimensional spaces and complex environments, it fails to guarantee optimal paths, exhibits low path search efficiency, and produces non-smooth paths [14].

A brief overview of intelligent algorithms follows. In 1992, Marco Dorigo invented the Ant Colony Optimization algorithm by simulating the foraging behavior of ants [15]. Despite its excellent global optimization capabilities, the ant colony algorithm tends to fall into local optima due to its computational intensity [16]. Based on the principles of birds seeking food, Kennedy and Eberhart proposed the Particle Swarm Optimization algorithm in 1995 [17]. However, this algorithm suffers from slow convergence in search depth and weak local search capabilities [18]. Genetic algorithms, introduced by John Holland in the 1970s, draw inspiration from the law of organized evolution, providing faster and better optimization results for more complex combinatorial optimization problems

*Corresponding Author: Qing-Hua Luo; E-mail: luoqinghua081519@163.com
DOI: https://doi.org/10.70003/160792642025032602004

[19]. Nevertheless, genetic algorithms exhibit poor local search capabilities, premature convergence, slow convergence speed, and the complexity of operations such as selection, mutation, and crossover, conducted probabilistically [20].

In addition to these, novel intelligent algorithms like the Grey Wolf Algorithm find widespread application in robot path planning due to their simplicity, few parameters, easy programming, support for distributed parallel computing, and robust global search capabilities [21]. Neural networks optimize AGV path planning and obstacle avoidance by considering environmental states and potential obstacles, thereby enhancing computational efficiency [22]. Reinforcement learning involves the continuous correction of strategies through the interaction between an agent and the environment to learn optimal action strategies [23]. However, it tends to face challenges in training time and sample efficiency, especially in complex scenarios. Therefore, combining deep learning with reinforcement learning addresses decision-making issues and enables efficient execution of path planning tasks for mobile robots [24]. Nevertheless, these methods have their drawbacks, such as the Grey Wolf Algorithm potentially getting trapped in local optima for complex problems, neural network path planning facing computational complexity and requiring substantial labeled data, reinforcement learning posing challenges in training time and sample efficiency, and deep reinforcement learning being susceptible to sample complexity and overfitting issues. These factors limit their performance and applicability in certain contexts.

In comparison to the algorithms, the A-star algorithm not only boasts simplicity in static global planning but also exhibits fast operation during path search, making it widely applicable in computer science and robotics research fields. For instance, in areas like robot navigation, the A-star algorithm effectively identifies the shortest path. In game development, the A-star algorithm can generate intelligent movement paths for NPCs. Furthermore, the A-star algorithm allows the flexibility to choose different heuristic functions for specific problems. However, the A-star algorithm has its drawbacks. The selection of the heuristic function significantly influences the algorithm's performance, and designing an accurate heuristic function is challenging, requiring domain knowledge and experience. Additionally, in complex environmental situations, the traditional A-star algorithm faces issues such as low search freedom and paths with numerous turning points during path planning.

Therefore, addressing the issue of excessive planning nodes and long paths in traditional A-star path planning, this paper proposes a heuristic-based path planning algorithm. Firstly, to mitigate the environmental impact on the algorithm, an adaptive algorithm is employed to analyze the characteristics of obstacles in the map, obtaining specific obstacle dimensions. Secondly, to reduce the iterative computation load and shorten the path length by minimizing path nodes, this paper connects the starting point and endpoint with a straight line, checks if the line intersects with obstacles, and identifies intersection points. If the straight line does not intersect with obstacles, it is considered the optimal path, eliminating the need for further iterative calculations, and the path length is inherently the shortest. Finally, we introduce search factors *Connecting_distance* and β in the A-star algorithm to enhance the algorithm's search neighborhood, thereby addressing the traditional algorithm's issue of suboptimal path planning due to low search freedom.

In summary, our proposed heuristic-based path planning algorithm improves exploration by simplifying the path calculation process, adapting to obstacle sizes, and enhancing heuristic functionality, resolving the issue of excessive path nodes and long paths in traditional A-star methods. The remaining sections of this paper are organized as follows. Section two presents relevant previous research. Section three outlines the overall concept and specific implementation steps of the algorithm. Section four evaluates the performance of the proposed algorithm and conducts a comparative analysis with related algorithms. Section five provides a summary of the entire paper and suggests future research directions.

2 Related Research

In recent years, with the continual advancement of computer technology and artificial intelligence, scholars across diverse domains have intensified their efforts in enhancing path planning techniques. For instance, reference [25] proposes a hybrid path planning algorithm based on the Membrane Pseudo-Bacterial Potential Field (MemPBPF), reducing time complexity through the integration of membrane computation, pseudo-bacterial genetic algorithm, and Artificial Potential Field (APF) methods. This achieves improved feasible solutions while considering minimum path length, collision avoidance, and path smoothness. Reference [26] introduces an enhanced Motion-constrained Bidirectional Rapidly Exploring Random Tree (IKB-RRT) algorithm, incorporating guided nodes under robot kinematic constraints. It heuristically guides the growth of the random expansion tree into the configuration space target, mitigating collisions with obstacles. To prevent abrupt changes in heading due to independent tree expansion leading to connection points, a dual-tree region path smoothing optimization connection strategy is proposed, enhancing the overall smoothness of the planned path. Reference [27] combines membrane computation with genetic algorithms and artificial potential field methods, seeking parameters to generate feasible and safe paths. Reference [28] addresses the slow convergence of Q-learning towards optimal solutions by incorporating the concept of partially guided Q-learning. It improves classical Q-learning using the APF method, enhancing learning speed and final performance.

In the realm of global planning algorithms, the A-star algorithm stands as a focal point of path planning research, with numerous scholars and research teams making significant contributions. They are dedicated to optimizing the performance of the A-star algorithm, expanding its applicability, and exploring various improvements and extensions related to the A-star algorithm. Reference [29], building upon the foundation of unit decomposition and map updating, devises an improved BA algorithm to address continuity deficiencies and high-precision environmental modeling challenges. Citing [6], the enhanced A-star algorithm is combined with Delaunay triangulation, presenting a dynamic fusion path planning algorithm. Delaunay triangulation is employed to handle complex obstacles, and it can also generate Voronoi points. Reference [4] introduces geometric A-star algorithms. To enhance the stability of AGV in turning paths, this paper employs functions to filter nodes in the closed list and replaces turning points with cubic B-spline curves. Reference [30] introduces a new heuristic function to enhance the performance of the A-star algorithm, incorporating not only distance information but also obstacle information.

Reference [31] proposes a method that prioritizes the expansion of adjacent nodes in the direction of endpoints. It incorporates turning costs into the calculation of the actual cost and estimated cost of the current node. Reference [32] improves the A-star algorithm by introducing an L-shaped path trend. As the A-star algorithm generates many turning points during path planning, this paper traverses turning nodes in the path and, in the absence of obstacles, smoothes the path by replacing the current turning node with a diagonal node forming a loop.

Reference [33] suggests a fusion approach of A-star and dynamic window methods, optimizing search angles and combining them with the path of the dynamic window method. The improved method exhibits good efficiency and feasibility. Reference [34] enhances the A-star algorithm by incorporating the vehicle's kinematic model as a constraint in the cost function. The obtained path is smoother, more rational, and aligns with the vehicle's kinematic model. Reference [35] improves the A-star algorithm by adding environmental information and AGV position information to the traditional evaluation function. The algorithm optimizes path points and eliminates unnecessary turning points.

Reference [36] introduces the guiding principles and key points of the A-star algorithm to develop a heuristic function, making it easier to avoid obstacles. Reference [37] improves the A-star algorithm by considering the distance factor between obstacles, preventing redundant nodes caused by being too close to obstacles. Reference [38] introduces criteria, key points, and a new variable step size into the A-star algorithm, reducing computation time.

Reference [41] combines the renowned hybrid A-star search engine with the "visibility map" project to obtain the optimal path. Reference [39] utilizes jump point search to optimize the search method and search speed. Simultaneously, this paper considers angle evaluation costs, leading to a shorter path. Reference [10] adjusts the number of directions extending from the current point to surrounding survey points. Reference [9] designs a novel heuristic algorithm, where the heuristic function uses energy consumption to estimate costs.

3 Materials and Methods

This section aims to intricately elucidate the innovative concepts and specific implementation steps undertaken to enhance the A-star algorithm. Initially, we provide a comprehensive overview of each constituent element of the proposed algorithm, delving into the specific details of each component. Finally, to present the algorithm's workflow more lucidly, we summarize the entire process and accompany it with a flowchart.

Our improvement approach necessitates, as a primary step, connecting the starting point and endpoint through a straight line. Recognizing that obstacles may not lie directly on this line, our algorithm explicitly considers the straight line as the optimal path, obviating the need for iterative calculations. Consequently, our focus here is specifically on scenarios where at least one obstacle is present.

3.1 The Overall Process of Our Proposed Algorithm

Figure 1 describes the overall process of the algorithm. It consists of four main parts: adaptively determine obstacle size, determine the handover point, improved the A-star algorithm, and combine local paths. We first compute the measurement of the obstacle through an adaptive algorithm, then make a circle with this radius, determine the handover point, use the improved A-star algorithm in each segment of the local path, and the local paths are finally merged.

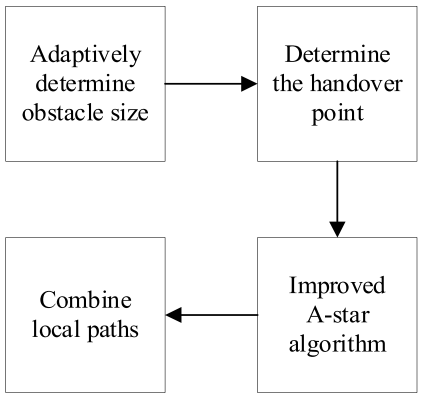

Adaptively determine obstacle size
Determine the handover point
Improved A-star algorithm
Combine local paths

Figure 1. The framework of the path planning based on a heuristic method

(1) Adaptively determine obstacle size: Use an adaptive algorithm to find the maximum boundary distance of a given obstacle. For regular graphics such as a rectangle, the maximum boundary distance is the rectangle diagonal.

(2) Determine the handover point: The straight path, between the start and the endpoint, will have two points of intersection with the circle centered on an obstacle. The handover point can be defined as follows: we draw a circle, and the circle radius is the size of the obstacle. The two intersections between the circle and the straight line are called the handover points.

(3) Improved A-star algorithm: We introduce the weighted factor β in the heuristic function part and

introduce the connection factor Connecting_distance that describes the direction in which the current point expands to the surroundings.

(4) Combine local paths: After determining the local start and the local endpoint, we use the improved A-star algorithm to generate the local path. And then we combine the local paths into a global path.

3.2 Adaptively Determine Obstacle Size

We determine the size of the obstacle by using an adaptive algorithm. And when we improve the A-star algorithm, we also need to consider the obstacle measurement.

$$M = \{m_i\} \quad (1)$$

Here, M is the map, divide the map into smaller grids, each grid is represented by m_i , m_i value of 1 means that there is an obstacle at that grid, otherwise, m_i value of 0 means that the grid is passable.

We assume a situation where an obstacle is composed of some grid cells, and the following expressions (2) and (3) are used to describe the obstacle M .

$$M = \sum_{j=1}^{n} m_j \quad (2)$$

$$\{m_i\} = 1, j = 1,2,3,\ldots,n \quad (3)$$

Figure 2. shows a square obstacle, each m_i of which is 1. We consider the case when there exists only one obstacle, and the obstacle is composed of many smaller obstacles. The specific implementation steps are listed in Algorithm 1. The input M of the algorithm in Algorithm 1 is an obstacle, and the output Max is the size of the obstacle.

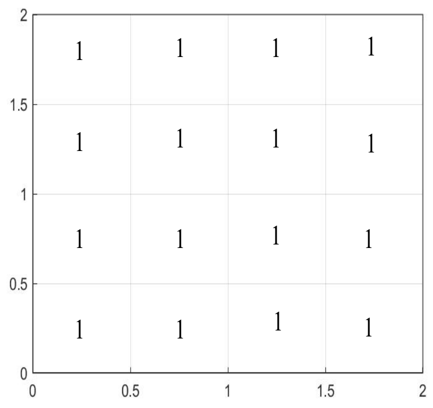


Figure 2. Representation of the obstacle

The proposed algorithm utilizes an adaptive algorithm to find the maximum boundary distance of a given obstacle. For regular graphics (such as rectangles), the maximum value is the length of the diagonal of the rectangle.

Algorithm 1. Size of obstacles

```
1  $Max = 0$
2  for $m_i$ in $M$ do
3   if $m_i$=1 then
4      $obstacle=[obstacle\ m_i]$
5   end if
6  end for
7  $n$=size($obstacle$)
8  for $i = 1$ to $n$ do
9   for $j = 1$ to $n$ do
10    $temp = |obstacle(i) - obstacle(j)|$
11    if $temp > Max$ and $i \neq j$ do
12       $Max = temp$
13    end if
14   end for
15 end for
16 return $Max$
```

3.3 Determine the Handover Point

In this situation, suppose there only exists one obstacle. We first plan a straight path from the start to the endpoint and show it in Figure 3. S is the start and its coordinates are (x_{start}, y_{start}) , E is the endpoint and its coordinates are (x_{goal}, y_{goal}) , O is the obstacle and its coordinates are $(x_{obstacles}, y_{obstacles})$. We draw a circle using O to be the center and intersect the straight-line SE at points M and N, the radius of the circle here is r.

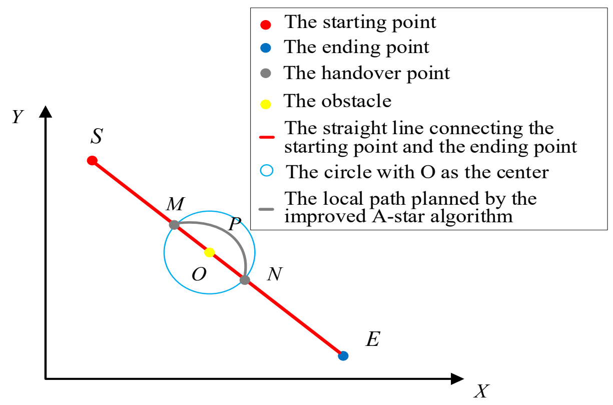

● The starting point
● The ending point
● The handover point
● The obstacle
The straight line connecting the starting point and the ending point
○ The circle with O as the center
The local path planned by the improved A-star algorithm
Y
X
S
M
P
O
N
E

Figure 3. Simulation diagram of path planning for our presented algorithm when considering one obstacle

The equation of the straight-line SE is as follows:

$$\frac{y - y_{start}}{y_{goal} - y_{start}} = \frac{x - x_{start}}{x_{goal} - x_{start}} \quad (4)$$

And the equation of the circle using O to be the center is shown below:

$$(x - x_{obstacle})^2 + (y - y_{obstacle})^2 = r^2 \quad (5)$$

We determined the coordinates of the handover points M and N by using the joint equations 4 and 5. And the coordinates are $M(x_M, y_M)$ and $N(x_N, y_N)$ respectively.

To bypass the detected obstacle O, we need to plan the local path by using M and N as the start and the endpoint, and we determine them by the following method:

$$x_M - x_N = D \qquad \textbf{(6)}$$

If D is larger than 0, which shows that point N is closer to the starting point. So N is chosen to be the local start and M is chosen to be the local endpoint. Otherwise, we make the opposite choice.

3.4 Improved A-star Algorithm

We introduce the weighted factor β in the heuristic function part to improve the A-star algorithm. It also introduces the connection factor Connecting_distance to describe the direction in which the current point expands to the surroundings.

The A-star algorithm first uses the current point as a reference point. Then it evaluates the score of the surrounding points. Finally, we select the point of the lowest score as the next point.

Therefore, the valuation of the location is very important, and the cost function is improved by the following method:

$$f(n) = g(n) + \beta h(n) \qquad \textbf{(7)}$$

Here n is the current point. $f(n)$ is the valuation function of point n. $g(n)$ being the minimum path cost between the start and the present point. β is a constant parameter, and it will be specifically optimized later. $h(n)$ being the minimum estimated path cost of the path between the present point and the endpoint.

We use Euclidean distance in the calculation of the estimated cost between the present point and the endpoint.

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2} \qquad \textbf{(8)}$$

Here x_n , x_{goal} are the horizontal coordinates of the present point and the endpoint respectively; y_n , y_{goal} are the vertical coordinates of the present point and the endpoint respectively.

The standard A-star algorithm will only search eight surrounding points, which may lead to sub-optimal paths. We define a parameter as connecting distance, when the connecting distance takes different values, the improved A-star algorithm will search for points in different number of directions.

More specifically, when Connecting_distance = 1, 2, 3 and 4, the algorithm will be 8, 16, 32, and 48 directions to expand around. To explain more clearly, we have drawn the diagrams of the Connecting_distance, and represent them in Figure 4. Table 1 shows how many expansion directions correspond to different values of the Connecting_distance.

Below, we will give the specific algorithm of the direction of the surrounding expansion generated by different Connecting_distance, as shown in Algorithm 2.

Table 1. The choice of options

Connecting_distance	Number of directions
1	8
2	16
3	32
4	48

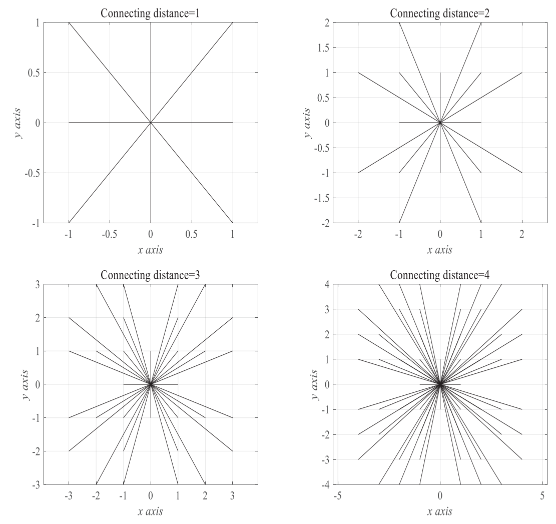


Figure 4. Using different connecting distance to expand the graph around

Algorithm 2. Path orientation

```
1 $NeighboorCheck = ones(2•Connecting\_distance+1)$
2 $Dummy = 2•Connecting\_distance+2$
3 $Mid = Connecting\_distance+1$
4 for $i=1$ to $Connecting\_distance-1$ do
5     $NeighboorCheck(i,i)=0$
6     $NeighboorCheck(Dummy-i,i)=0$
7     $NeighboorCheck(i,Dummy-i)=0$
8     $NeighboorCheck(Dummy-i,Dummy-i)=0$
9     $NeighboorCheck(Mid,i)=0$
10    $NeighboorCheck(Mid,Dummy-i)=0$
11    $NeighboorCheck(i,Mid)=0$
12    $NeighboorCheck(Dummy-i,Mid)=0$
13 end for
14 $NeighboorCheck(Mid,Mid)=0$
15 $[row,col]=find(NeighboorCheck==1)$
16 $Neighboors=[row,col]-(Connecting\_distance)$
17 return $Neighboors$
```

Finally, we combine the given weighted factor β in the heuristic function and path orientation analysis obtained above to better the performance of the A-star algorithm. And give the flow chart as follows.

3.5 Combine Local Paths

From Figure 3, we can know that the start and the endpoint of the local path are *M* and *N* respectively. Then we use the presented A-star algorithm in this paper for the local path and combine it with the straight path to obtain the complete path [*SMPNE*].

For the most part, there will be multiple obstacles in the straight-line *SE*. Take the example of Figure 5, which contains two obstacles O_1 and O_2. Following the method described in the previous section, we obtain the path [$SM_1P_1N_1M_2P_2N_2E$]. However, to further optimize the path and shorten the path length, we proposed a method to merge local paths. When the distance between two obstacles is close, use the start of the previous pair of handover points as the start, and use the endpoint of the latter pair of handover points as the endpoint. To illustrate through Figure 5. is to combine [x_{M1} , x_{N1}] and [x_{M2} , x_{N2}] into [x_{M2} , x_{N2}]. To ensure that the merged path length becomes shorter, we will evaluate the threshold to determine a suitable threshold value later. We draw a circle with the endpoint of the previous local path as the center and the threshold as the radius, if the start of the latter local path is inside the circle, the two paths are combined, otherwise not merged. Specific ways to achieve as follows.

$$(x_{N_1}-x_{M_2})^2+(y_{N_1}-y_{M_2})^2 \le R^2_{threshold} \tag{9}$$

When the two obstacles in Figure 5. satisfy the condition, we obtain the path [$SM_1P_3N_2E$].

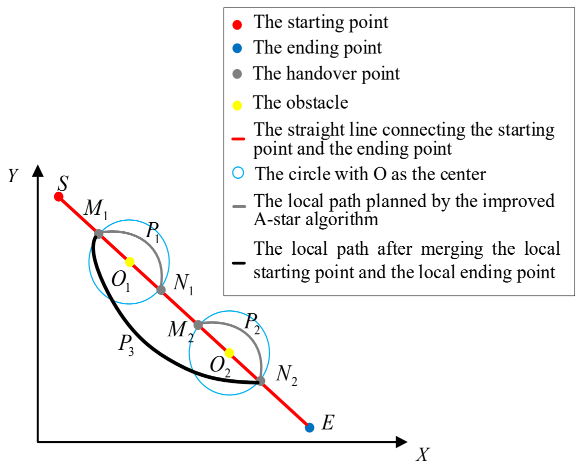

● The starting point
● The ending point
● The handover point
● The obstacle
The straight line connecting the starting point and the ending point
○ The circle with O as the center
The local path planned by the improved A-star algorithm
The local path after merging the local starting point and the local ending point
Y
X
S
M1
P1
O1
N1
M2
P2
O2
N2
P3
E

Figure 5. Simulation diagram of path planning for our presented algorithm when considering two obstacles

3.6 Summary of the Proposed Algorithm

We have described the principle of each part of our algorithm in the previous sections. Figure 6 illustrates the basic flow of the A-star algorithm. Figure 7, shows how they work.

As shown in Figure 7, we first connect the start and endpoint. If the straight path from the start to the endpoint does not pass the obstacles. Our proposed algorithm will not perform iterative calculations. Instead, it will directly take the straight line as the optimal path.

In this section, we assume that there is at least one obstacle between the start and endpoint. We first use the algorithm in Algorithm 1 to determine the size of obstacles adaptively. Then we determine the handover point. After that, we introduce the weighted factor *β* and the connection factor Connecting_distance for the A-star algorithm. The algorithm for determining the number of search directions to the surroundings according to Connecting_distance is shown in the algorithm in Algorithm 2. Finally, we combine local paths into global paths.

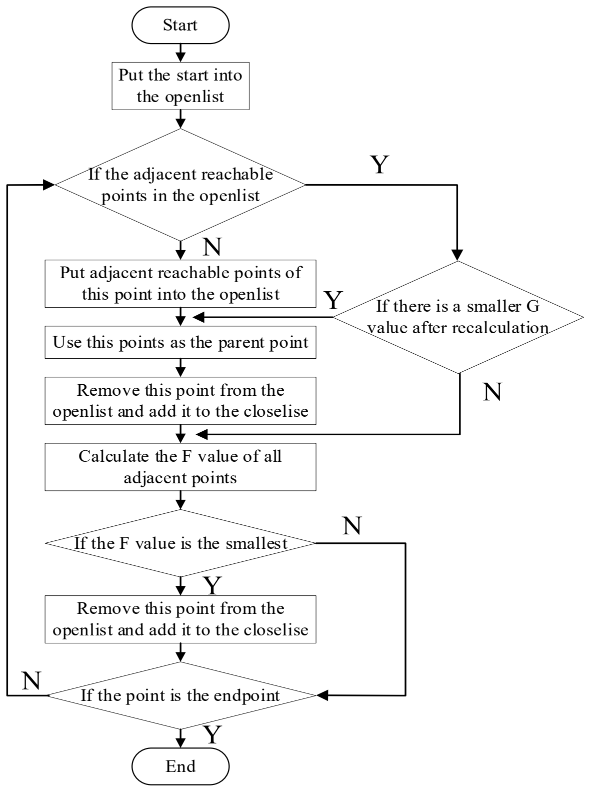

Start
Put the start into the openlist
If the adjacent reachable points in the openlist
Y
N
Put adjacent reachable points of this point into the openlist
Y
If there is a smaller G value after recalculation
Use this points as the parent point
Remove this point from the openlist and add it to the closelise
N
Calculate the F value of all adjacent points
If the F value is the smallest
N
Y
Remove this point from the openlist and add it to the closelise
N
If the point is the endpoint
Y
End

Figure 6. The improved A-star algorithm flow chart

To explain the proposed algorithm more clearly, we give the flowchart of the algorithm in Figure 7. Furthermore, we next describe the algorithm steps in detail.

Step 1: We calculate the equation of the straight-line *SE*.

Step 2: We check whether the straight-line *SE* passes through the obstacle. If not, the optimal path is the straight-line *SE*, and exit the algorithm; otherwise, skip to Step 3.

Step 3: There are *n* obstacles crossed on the straight line, which is marked as O_i (i = 1, 2, ..., n). Treat each obstacle as follows:

(a) We regard the obstacle O_i as the center and *r* as the radius, then we calculate the equation of the circle.

(b) We calculate the intersections *M* and *N* of the circle in (a) and the straight-line *SE*, and then determine the start and the endpoint according to equations 6, denoted as $local\_start(i)$ and $local\_goal(i)$.

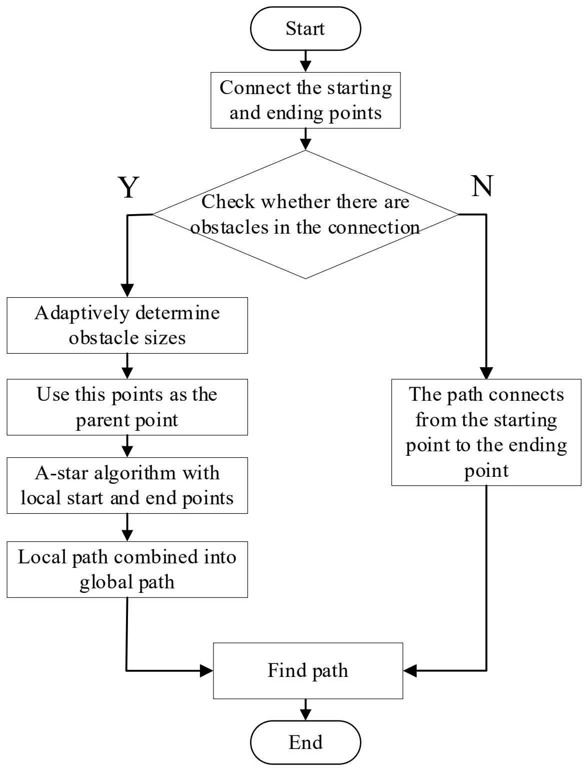
Step 4: The start and the endpoint of each local line segment obtained in Step 3 are optimized as follows:

If the $|local\_goal(i)-local\_start(i+1)|\le\varepsilon$ is satisfied, the $|local\_goal(i), local\_start(i)|$ and the $|local\_goal(i+1), local\_start(i+1)|$ can be merged into $|local\_goal(i), local\_start(i)|$. Otherwise, it will remain unchanged.

Step 5: Perform the following operations on each pair of the start and the endpoint obtained after optimization in Step 4:

The starting point *local_start*(*i*) and the endpoint *local_goal*(*i*) are substituted into the improved A-star algorithm, a path is generated and recorded as *path*(*j*), *j*=1, 2, 3, ..., *n* and $j\le n$.

Step 6: We combine the path generated in Step 5 and the path of the straight-line *SE*. The final path point is [*S*, *path*(1), *path*(2), ... , *E*], where *j*=1, 2, 3, ..., *n* and $j\le n$.



Start
Connect the starting and ending points
Y
N
Check whether there are obstacles in the connection
Adaptively determine obstacle sizes
Use this points as the parent point
A-star algorithm with local start and end points
Local path combined into global path
The path connects from the starting point to the ending point
Find path
End

Figure 7. Path planning based on a heuristic algorithm flow chart

4 Results

In this segment, we endeavor to assess the algorithm proposed within this manuscript across diverse experimental scenarios. The considerations encompass the following instances:

(1) Unimpeded traversal along a direct trajectory from the initiation point to the terminal point.

(2) The linear path from the commencement to the conclusion marred by the presence of a mere two obstacles.

(3) The linear trajectory from commencement to conclusion adorned with an array of more than two obstacles.

(4) The existence of serendipitous impediments interspersed between the initiation and terminal points.

Our initial course of action involves the meticulous configuration of the experimental milieu. Subsequently, a judicious adjustment of parameters within the proposed algorithm ensues to optimize outcomes. Finally, we conduct simulated experiments, thereby garnering pertinent data for both the proposed methodology and the benchmark methods.

4.1 Experimental Setup

To rigorously assess the efficacy of the proposed algorithm, this section orchestrates the setting of the robot's path planning map environment. It introduces the evaluation metrics for algorithm assessment, elucidates the four comparative algorithms against our refined approach.

4.1.1 Map and Experiment Environment Setting Up

In this segment, we scrutinize the performance of our proposed algorithm within a static grid environment.

Test 1: Initially, we delineate a grid map measuring 140 units in length and 120 units in width. Grid values of 1 correspond to obstacles, while values of 0 denote obstacle-free areas. The starting point is set at (20,10), with the endpoint at (70,110).

Test 2: To validate the effectiveness of our improvements to the A-star algorithm, we opt for the map environments utilized in references [42] and [43]. The map construction, starting point, and endpoint configurations align with those specified in the aforementioned references. To enhance the credibility of the enhanced A-star algorithm, we employ a map environment of dimensions 200×200, introducing randomized dynamic obstacles. Refer to Table 2 for detailed parameters regarding the map setup, where a 500×500 scale map represents the environments described in [42] and [43].

Table 2. Map parameters

Map scale	Starting point	Endpoint
500×500	(50, 250)	(400, 200)
	(50, 425)	(400, 225)
200×200	(25, 35)	(140, 190)

CPU: Intel^{TM} core i7 8750H CPU@2.20GHz, 8G RAM, Windows 10 64bit.

4.1.2 Evaluation Metric

We propose two evaluation metrics to measure the algorithms used in the experiments:

$$R_{length}=\frac{length_{proposed}-length_{reference}}{length_{proposed}} \tag{10}$$

$$R_{time}=\frac{time_{proposed}-time_{reference}}{time_{proposed}} \tag{11}$$

Here $length_{reference}$ and $length_{proposed}$ represent the path length planned by our presented algorithm in this paper and the reference algorithm, respectively. $time_{reference}$ and $time_{proposed}$ represent the required time of the path planned by our presented algorithm and the reference algorithm, respectively. R_{length} and R_{time} show the improvement level of our presented algorithm in path length and required time.

The sign of R here indicates a certain significance. Take R_{time} as an example, if $R_{time} > 0$, we can know that the time required by our presented algorithm is $100 \times R_{time}$% longer than the reference algorithm, otherwise, the time required by our presented algorithm is $100 \times R_{time}$% shorter than the reference algorithm.

4.1.3 Reference Algorithm

We employed four reference algorithms, namely the traditional A-star algorithm (TA-star), the algorithm proposed in reference [40] (RA-star1), the method introduced in reference [41] (RA-star2), and the approach presented in literature [44] (CAPSO). The improved A-star algorithm in this paper is denoted as PA-star. Both algorithms from references [40] and [41] have refined the heuristic function within the A-star algorithm. To comprehensively assess the superiority of the enhanced A-star algorithm, we further chose to compare it with the improved particle swarm optimization algorithm proposed in literature [44], and meticulously replicated the algorithm for a fair evaluation. The assessment will be conducted using the aforementioned evaluation metrics. We will evaluate them through the evaluation metrics R_{length} and R_{time} presented above.

Reference [40] defines a new method to estimate the cost between the present point and the endpoint.

The heuristic function $f(n)$ represents are as below:

$$f(n) = g(n) + H(n) \tag{12}$$

The function $H(n)$ represents is defined as follows:

$$H(n) = (H\_sum - 2 \times H\_\min) + \sqrt{2} \times H\_\min \tag{13}$$

$$H\_\min = \min\{| x(s) - x(e), y(s) - y(e) |\} \tag{14}$$

$$H\_sum = | x(s) - x(e) | + | y(s) - y(e) | \tag{15}$$

Here the present point is $(x(s), y(s))$ and the endpoint is $(x(s), y(s))$. H_sum and $H\_\min$ are the Manhattan distance and the shortest of the horizontal distances and vertical distances from the present point to the endpoint respectively.

Reference [41] improves the idea by proposing an improvement A-star algorithm, which assigns a weighting factor β to the heuristic function to obtain a new heuristic function $f(n)$.

$$f(n) = g(n) + \beta h(n) \tag{16}$$

Here β can take a value between 1.5 and 2.

The method proposed in reference [44] introduces a path planning approach based on cubic spline interpolation. Selecting several path nodes as control points, cubic spline interpolation is applied to interpolate the path between the starting point, control points, and the target point, forming a complete trajectory. The newly introduced chaotic adaptive particle swarm optimization algorithm (CAPSO) is utilized to optimize the control points in the cubic spline interpolation.

To facilitate a fair comparison of their path planning performance, we essentially implemented the methods proposed in the references.

4.2 Parameter Selection

This section is dedicated to evaluating the three parameters in the algorithm proposed in this paper and the control points in CAPSO.

4.2.1 Parameter Evaluation

In the static path planning algorithm based on the heuristic approach proposed in this paper, three parameters significantly impact the effectiveness of path planning. These parameters are β in equation (7), the connection distance parameter *Connecting_distance*, and the radius r in equation (5). The parameter optimization method utilized in this study follows the approach used in literature [45]. Specifically, a diverse set of parameter combinations is established within the range of parameter variations. The optimal combination is then determined through statistical analysis of experimental results. This paper adopts the default parameter combination: $\beta = 1$, *Connecting_distance* = 10, $r = 1$. Simulation experiments are conducted in multiple obstacle environments, using the default parameter combination as a baseline. Each parameter value is altered individually, and to mitigate occasional variations, 10 experiments are performed, and the results are averaged. Through experimental analysis, the three parameter values that yield the best path planning performance are selected.

(1) Evaluation of Parameter β

The impact of varying parameter β on the path planning algorithm based on the heuristic approach proposed in this paper is assessed. The optimization results for parameter β are presented in Table 3. Figure 8 illustrates the influence of parameter β on the path planning algorithm proposed in this paper.

Table 3. The choice of options

Interval	Path length
[0 1.93]	115.76
[1.93,2.41]	116.58
[2.41 3.19]	119.07
[3.19 5]	119.9

From Figure 8 and Table 3, it is evident that, within the interval [0 1.93], the algorithm's generated path is shortest, measuring 115.76. As the value of parameter β increases, the path length also increases. Considering all factors, the value of parameter β in this paper is set to 1.

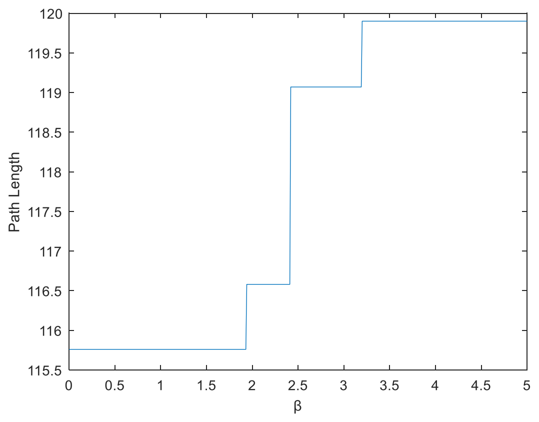


Figure 8. Influence of Parameter *β* on the path planning algorithm

(2) Evaluation of Parameter *Connecting_distance*

Initially, various values for the connection factor *Connecting_distance* are considered, with parameter *β* set to 1 and parameter r set to 10. The algorithm's planned path lengths are then assessed, and the results are shown in Figure 9.

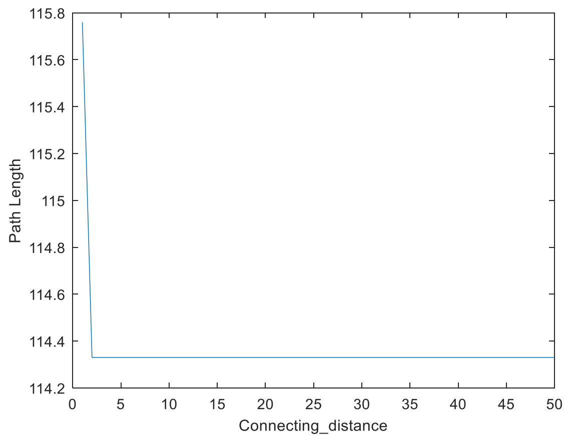


Figure 9. Influence of Parameter *Connecting_distance* on the path planning algorithm

From Figure 9, it can be observed that when *Connecting_distance* = 1, the path length is 115.76, and as *Connecting_distance* increases, the corresponding path length decreases. However, after *Connecting_distance* reaches 2, the path length stabilizes at a constant minimum value of 114.33. Considering the preference for a shorter path length, the value of parameter *Connecting_distance* in this paper is set to 2.

(3) Evaluation of Parameter *r*

Different values for the factor r are set, with parameter *β* set to 1 and parameter *Connecting_distance* set to 1. The algorithm's planned path lengths are then assessed, and the results are shown in Figure 10, with corresponding data in Table 4.

From Figure 10 and Table 4, it is evident that the shortest length, 115.56, occurs in the interval [1.4 5.0]. Considering the preference for a shorter path length, the value of 4 is selected for this parameter.

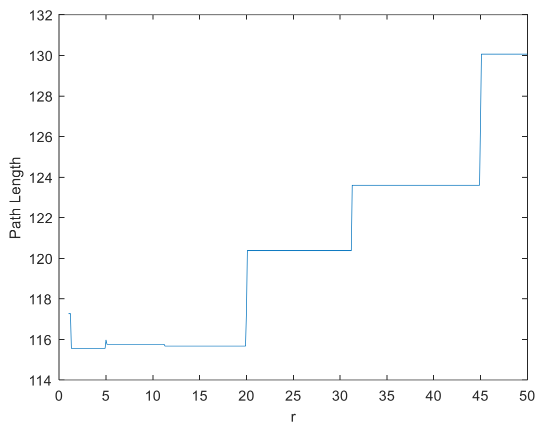


Figure 10. Result of optimizing parameter *r*

Table 4. Raw data of optimized parameter results

Interval	Path length
[1.0 1.3]	117.27
[1.4 5.0]	115.56
[5.1 11.1]	115.76
[11.2 19.9]	115.67
[20.0 31.2]	120.38
[31.3 44.9]	123.6
[45.0 50]	130.06

The different values for the three parameters in this paper's algorithm are shown in Table 5.

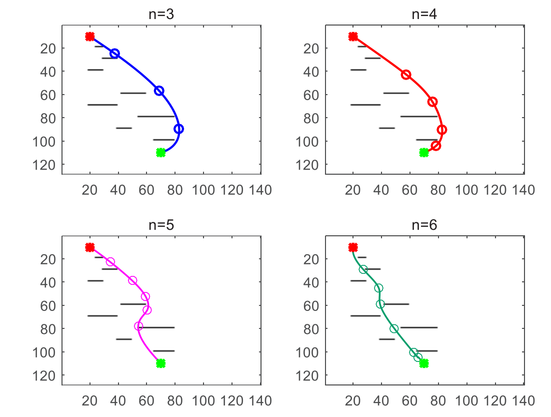
Table 5. Raw data of optimized parameter results

Parameter	Value
β	1.0
Connecting_distance	2.0
r	4.0

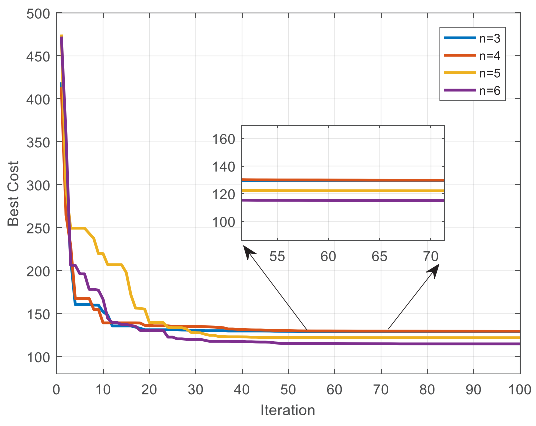
4.2.2 Control Point Selection

Literature [44] indicates that the quantity of control points corresponding to the CAPSO algorithm affects the path's effectiveness. Experimental results suggest that three to six control points are optimal. Therefore, this paper sets a population size of 30 and iterates 100 times to select the number of control points. Simulation experiments are conducted in multiple obstacle environments, and this experiment is performed 50 times to obtain averaged results. The results are depicted in Figure 11, where Subfigure (a) illustrates the paths planned by the CAPSO algorithm under different control points, and Subfigure (b) represents the iteration curves under different control points. The results indicate that the best outcome is achieved when there are 6 control points.

Hence, this paper sets the number of control points to 6, the population size to 30, the iteration count to 100, and other parameters consistent with literature [44].



(a) Simulation paths under different control points



(b) Iterative curve

Figure 11. Path of different control points in CAPSO

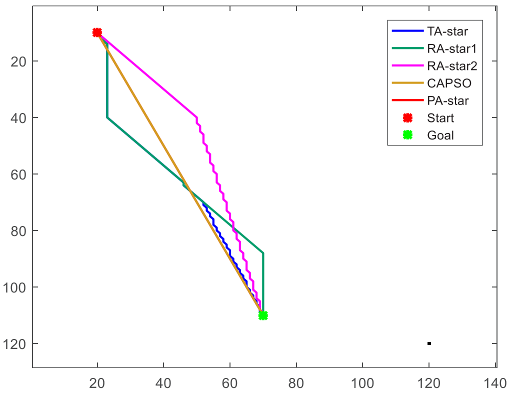


Figure 12. Paths of five algorithms without obstacles

Table 6. Path length without obstacles

Algorithm	Path length	R_{legth}
TA-star	120.7107	-7.9%
RA-star1	120.7107	-7.9%
RA-star2	120.7107	-7.9%
CAPSO	111.8404	0%
PA-star	111.8404	-

4.3 Comparison of PA-star with Four Reference Algorithms Regarding Pathways

Simulation is conducted 50 times for each map scenario in this study. To delve into a more comprehensive comparative analysis, both parametric and non-parametric statistical tests are employed. Given the fixed map scenarios, the paths obtained by each algorithm in each simulation experiment remain constant, and consequently, the path lengths are fixed. Thus, this study exclusively focuses on the statistical analysis of path search times.

4.3.1 Analysis of Unobstructed Paths between the Start and End Points

Figure 12 illustrates the paths obtained by the five algorithms in scenarios without obstacles. From Figure 12, it is evident that the paths generated by TA-star, RA-star1, and RA-star2 are longer than those planned by PA-star, while CAPSO produces paths identical to PA-star in obstacle-free environments. As depicted in Table 6, compared to TA-star, RA-star1, and RA-star2, PA-star results in a reduction of path lengths by 7.9%, 7.9%, and 7.9%, respectively.

4.3.2 Analysis of Two Obstacle Paths on the Line Connecting the Starting and Ending Points

We assume the straight path from the start to the endpoint pass only two obstacles. The following two situations need to be discussed:

When the parameter $R_{threshold}$ satisfies formula (9), we merge the two local paths, otherwise, it remained unchanged.

We tested in our experiments the effect of different values on the path length planned by our presented algorithm. We will illustrate the length of the path in Figure 13. From Figure 13, **when** $R_{threshold} < 13$, the path length is 124. When $R_{threshold} \geq 13$, the path length is 118.5.

From Figure 14, when $R_{threshold} < 13$, the two obstacles are not merged; when $R_{threshold} \geq 13$, the two obstacles are merged. So, it explains that the path length is shorter when $R_{threshold} \geq 13$. This article sets $R_{threshold} = 13$.

Figure 15 depicts the paths obtained by the five algorithms in scenarios with two obstacles. It is evident from Figure 15 that all five algorithms successfully find complete paths. However, paths generated by TA-star, RA-star1, and RA-star2 are longer than those planned by PA-star, while CAPSO demonstrates superior path planning in environments with two obstacles. As shown in Table 7, compared to TA-star, RA-star1, and RA-star2, PA-star yields path length reductions of 2.81%, 2.81%, and 2.81%, respectively. Additionally, when compared to CAPSO, PA-star exhibits a 4.77% increase in path length.

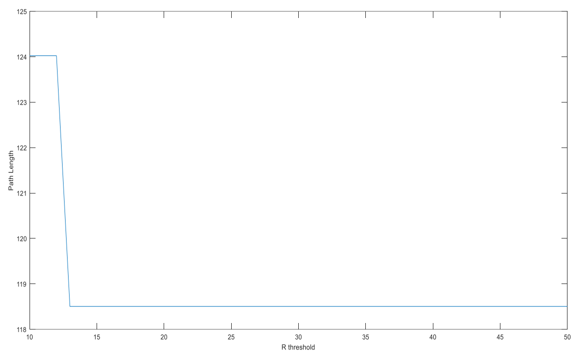
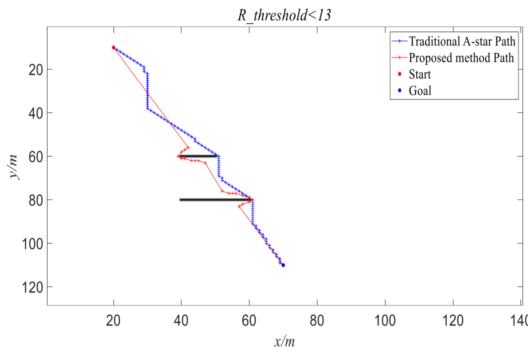
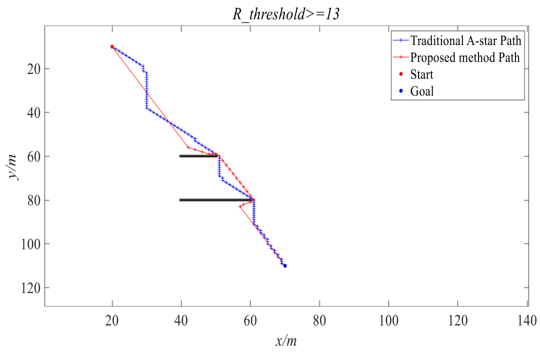


Figure 13. Result of optimizing parameter $R_{threshold}$



(a) Path planning based on our presented algorithm when $R_{threshold} < 13$



(b) Path planning based on our presented algorithm when $R_{threshold} \geq 13$

Figure 14. The path planned by our presented algorithm when there are different $R_{threshold}$

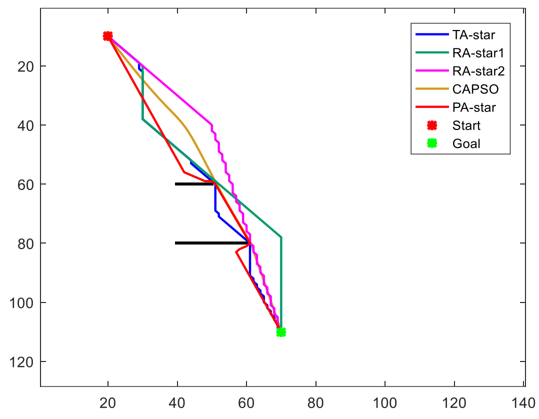


Figure 15. Paths of five algorithms under two obstacles

Table 7. Path lengths under two obstacles

Algorithm	Path length	R_{legth}
TA-star	121.83	-2.81%
RA-star1	121.83	-2.81%
RA-star2	121.83	-2.81%
CAPSO	112.85	4.77%
PA-star	118.50	-

4.3.3 Analysis of Multiple Obstacle Paths on the Line Connecting the Starting and Ending Points

Figure 16 illustrates the paths obtained by the five algorithms in scenarios with multiple obstacles along the line between the starting and ending points. It is evident from Figure 16 that all five algorithms successfully find complete paths. However, paths generated by TA-star, RA-star1, and RA-star2 are longer than those planned by PA-star, while CAPSO demonstrates superior path planning in environments with multiple obstacles. As shown in Table 8, compared to TA-star, RA-star1, and RA-star2, PA-star yields path length reductions of 5.27%, 5.27%, and 12.58%, respectively. Additionally, when compared to CAPSO, PA-star exhibits a marginal 0.18% increase in path length.

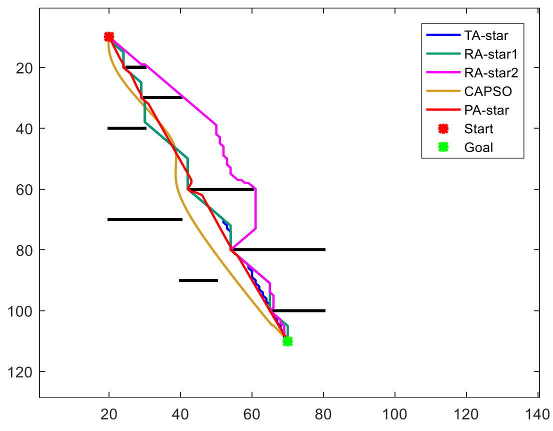


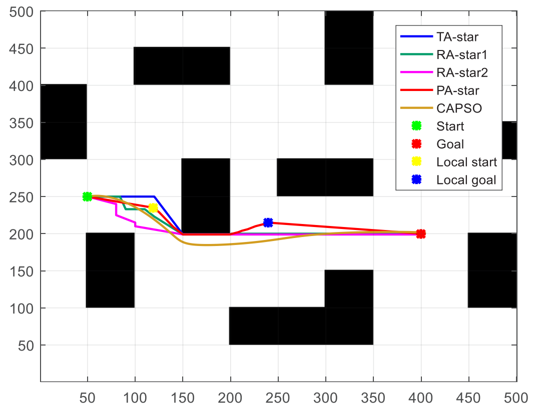
Figure 16. Paths of five algorithms under multiple obstacles

Table 8. Path length under multiple obstacles

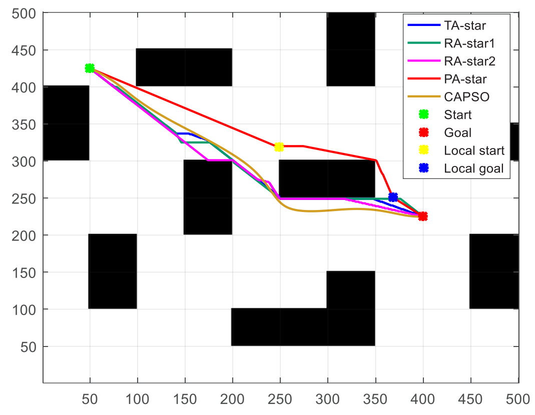
Algorithm	Path length	R_{legth}
TA-star	120.71	-5.27%
RA-star1	120.71	-5.27%
RA-star2	129.09	-12.58%
CAPSO	114.46	0.18%
PA-star	114.67	-

4.3.4 500×500 Map Simulation Comparison

Figure 17 illustrates the paths obtained by the five algorithms in a map scenario of size 500×500. Table 9 presents the path lengths obtained by the five algorithms in the 500×500 map scenario. From Figure 17, it is evident that PA-star can obtain specific paths in complex scenarios, where yellow points represent locally computed starting points by PA-star, blue points represent locally computed ending points, and the paths between local starting and ending points are planned using the improved A-star algorithm, while the other parts of the paths consist of straight lines. According to the data in Table 9, in scenario 1, compared to TA-star, RA-star1, RA-star2, and CAPSO, PA-star reduces the path length by 1.7%, 1.03%, 1.24%, and 2.98%, respectively. In scenario 2, compared to TA-star, RA-star1, RA-star2, and CAPSO, PA-star reduces the path length by 2.11%, 2.09%, 3.21%, and 0.53%, respectively. This demonstrates that PA-star can effectively obtain excellent paths in complex scenarios.



(a) Scenario 1



(b) Scenario 2

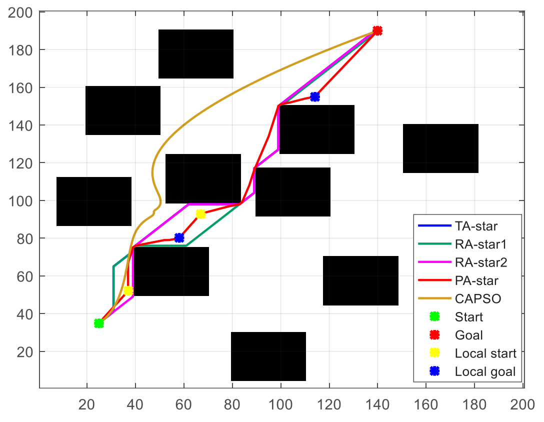
Figure 17. Simulation results in a 500×500 scale scenario

Table 9. Path length in 500×500 scale scenarios

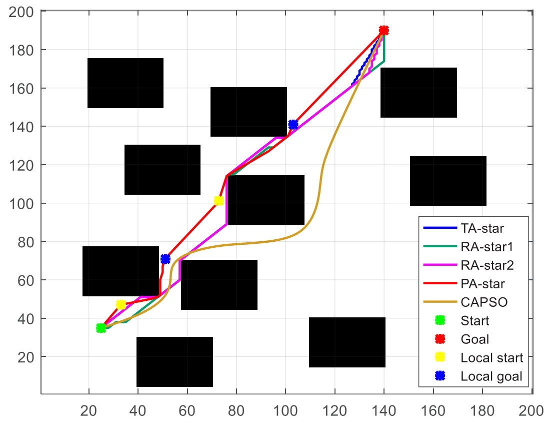
MAP	Algorithm	Path length	R_{legth}
Scenario 1	TA-star	377.58	-1.70%
	RA-star1	375.11	-1.03%
	RA-star2	375.88	-1.24%
	CAPSO	382.34	-2.98%
	PA-star	371.27	-
Scenario 2	TA-star	434.01	-2.11%
	RA-star1	433.94	-2.09%
	RA-star2	438.70	-3.21%
	CAPSO	427.30	-0.53%
	PA-star	425.06	-

4.3.5 200×200 Map Simulation Comparison

Figure 18 represents the paths obtained by the five algorithms in a map scenario of size 200×200. Table 10 presents the path lengths obtained by the five algorithms in the 200×200 map scenario. From Figure 18, it can be observed that PA-star can find a complete path even in a random obstacle map where there are multiple obstacles between the starting and ending points. Due to the close proximity of some obstacles, local starting and ending points are calculated with a distance smaller than $R_{threshold}$, and therefore, the paths are merged. According to the data in Table 10, in scenario 1, compared to TA-star, RA-star1, RA-star2, and CAPSO, PA-star reduces the path length by 2.27%, 2.27%, 2.27%, and 0.07%, respectively. In scenario 2, compared to TA-star, RA-star1, RA-star2, and CAPSO, PA-star reduces the path length by 3.69%, 3.69%, 3.69%, and 6.68%, respectively.



(a) Scenario 1



(b) Scenario 2

Figure 18. Simulation results in a 200×200 scale scenario

Table 10. Path length in 200×200 scale scenarios

MAP	Algorithm	Path length	R_{legth}
Scenario 1	TA-star	215.52	-2.27%
	RA-star1	215.52	-2.27%
	RA-star2	215.52	-2.27%
	CAPSO	210.89	-0.07%
	PA-star	210.74	-
Scenario 2	TA-star	209.07	-3.69%
	RA-star1	209.07	-3.69%
	RA-star2	209.07	-3.69%
	CAPSO	215.09	-6.68%
	PA-star	201.63	-

In these scenarios, it is evident that the paths planned by PA-star are superior to those of TA-star, RA-star1, and RA-star2. It is worth noting that the CAPSO algorithm tends to prefer areas with sparse obstacles during path planning because control points are more likely to be distributed in those regions. This emphasizes that CAPSO may not achieve optimal path planning in obstacle-dense maps. In contrast, PA-star consistently demonstrates outstanding performance in various map scenarios. While CAPSO might obtain shorter paths in maps with sparse obstacles, PA-star consistently achieves superior paths in obstacle-dense maps. This highlights the consistent quality of paths generated by PA-star in various map scenarios, showcasing its comprehensive advantages in path planning.

4.4 Comparison of Time between PA-star and Four Reference Algorithms

This study conducted 50 simulation experiments for each algorithm in different scenarios, recording the path search time in each iteration. A "Shapiro-Wilk" normality test was performed on the data, and the results showed a p-value less than 0.05, indicating a deviation from normal distribution. Considering the non-normal distribution of the data, non-parametric methods were chosen for further analysis. To compare differences between groups, the Kruskal-Wallis test was applied. The test statistics are presented in Table 11.

Table 11. Test statistic

MAP	Chi-square	DF	p
No obstacles	241.19	4	5.13e-51
Two obstacles	239.19	4	1.38e-50
Multiple obstacles	239.14	4	1.42e-50
500×500 (1)	233.36	4	2.49e-49
500×500 (2)	230.17	4	1.21e-48
200×200 (1)	231.79	4	5.41e-49
200×200 (2)	239.37	4	1.27e-50

The substantial values of the test statistic (chi-square) in various scenarios, ranging from 233.36 to 241.19, accompanied by p-values much smaller than the significance level (all below 0.05), indicate significant differences in search times among the five algorithms in different scenarios.

For a more detailed exploration of inter-group differences, post hoc analysis using Dunn's method was conducted. Table 12 to Table 14 display the median search times (interquartile range) for PA-star compared to other algorithms, along with Z-values and p-values. According to the data in Table 12, PA-star exhibits significant differences in search times compared to the four reference algorithms in all three map scenarios ($p < 0.05$). In obstacle-free scenarios, PA-star achieves almost instantaneous results due to directly considering the straight line between the start and end points as the final path. However, in scenarios with multiple obstacles, the need to determine obstacle sizes and use the improved A-star algorithm to navigate around obstacles results in longer algorithm runtimes. RA-star2 algorithm performs relatively quickly in all three scenarios, followed by TA-star. Nevertheless, both RA-star2 and TA-star yield longer path lengths. CAPSO algorithm, although capable of obtaining shorter paths in sparse scenarios, requires significant time for path acquisition. RA-star1 exhibits longer path times compared to PA-star.

Table 12. Results of the temporal comparative analysis

MAP	Algorithm	Search time	Z	p
There are no obstacles	TA-star	0.21 (0.2~0.22)	6.94	3.77e-11
	RA-star1	0.78 (0.77~0.80)	10.41	2.05e-24
	RA-star2	0.037 (0.037~0.041)	3.47	0.0052
	CAPSO	4.68 (4.679~4.68)	13.89	7.20e-43
	PA-star	0 (0~0)	-	-
Two obstacles	TA-star	0.23 (0.21~0.23)	3.45	0.0054
	RA-star1	0.89 (0.87~0.89)	-3.45	0.0054
	RA-star2	0.05 (0.045~0.05)	6.91	4.63e-11
	CAPSO	9.26 (9.23~9.31)	-6.91	4.63e-11
	PA-star	0.66 (0.65~0.66)	-	-
Multiple obstacles	TA-star	0.23 (0.22~0.24)	6.91	4.65e-11
	RA-star1	0.88 (0.86~0.89)	3.45	0.0054
	RA-star2	0.065 (0.061~0.068)	10.37	3.27e-24
	CAPSO	9.99 (9.96~9.06)	-3.45	0.0054
	PA-star	1.07 (1.06~1.09)	-	-

Table 13 indicates that in the 500×500 map, PA-star significantly differs in search times compared to the four reference algorithms ($p < 0.05$). RA-star2 demonstrates the fastest search times, followed by PA-star. However, RAstar2 yields relatively longer path lengths, showcasing PA-star's superior path search performance.

Table 13. 500×500 map-time comparison

MAP	Algorithm	Search time	Z	p
Scenario 1	TA-star	1.55 (1.51~1.61)	-3.14	0.016
	RA-star1	1.84 (1.82~1.89)	-6.47	9.94e-10
	RA-star2	0.74 (0.71~0.78)	3.84	1.24e-3
	CAPSO	8.78 (8.75~8.86)	-9.99	1.66e-22
	PA-star	1.31 (1.27~1.36)	-	-
Scenario 2	TA-star	2.03 (1.91~2.11)	-6.06	1.36e-8
	RA-star1	1.89 (1.87~1.91)	-4.31	1.62e-4
	RA-star2	0.83 (0.81~0.84)	3.45	0.0054
	CAPSO	8.75 (8.73~8.78)	-10.37	3.32e-24
	PA-star	1.56 (1.55~1.57)	-	-

Table 14. 200×200 map-time comparison

MAP	Algorithm	Search time	Z	p
Scenario 1	TA-star	0.29 (0.28~0.29)	6.24	4.29e-29
	RA-star1	0.94 (0.93~0.94)	2.10	0.349
	RA-star2	0.0657 (0.063~0.067)	9.70	2.95e-21
	CAPSO	8.92 (8.89~8.96)	-4.13	3.56e-4
	PA-star	1.04 (1.03~1.07)	-	-
Scenario 2	TA-star	0.314 (0.31~0.33)	6.91	4.53e-11
	RA-star1	1.19 (1.02~1.19)	3.45	0.0054
	RA-star2	0.057 (0.055~0.057)	10.37	3.10e-24
	CAPSO	8.92 (8.89~8.97)	-3.46	0.0054
	PA-star	1.76 (1.75~1.78)	-	-

According to the data in Table 14, in the 200×200 map, in scenario 1, the p-value for search time comparison between PA-star and RA-star1 is greater than 0.05 ($p = 0.349$), suggesting no significant difference in search times in scenario 1. In all other comparisons, PA-star shows significant differences in search times compared to the four reference algorithms ($p < 0.05$). PA-star's algorithmic search times in scenario 1 are 1.04 and 1.76, respectively. It's worth noting that in Figure 16, with multiple obstacles between the start and end points, determining obstacle sizes and using the PA-star algorithm skillfully to navigate around obstacles is necessary. Although search times increase for PA-star in this map, they remain smaller than CAPSO.

In conclusion, in complex environments, PA-star yields the shortest path lengths. This indicates that despite potentially requiring more time to handle intricate terrains, the PA-star algorithm demonstrates outstanding performance in path optimization, providing robust support for achieving the shortest paths. Therefore, the PA-star algorithm presents significant advantages in practical applications, especially in scenarios demanding precise and efficient path planning, offering users more accurate and efficient solutions.

5 Conclusion

In addressing the challenge of robot path planning, this study holistically considers the real-time and feasibility demands in practical applications, presenting an enhanced A-star algorithm. By judiciously integrating environmental information, this method effectively shortens path lengths. Simulation results demonstrate that, in terms of reducing path lengths, the improved A-star algorithm surpasses TA-star, RA-star1, and RA-star2 algorithms by at least 1.67%, 1.02%, and 1.23%, respectively. In dense scenarios, compared to the CAPSO algorithm, the enhanced A-star algorithm enhances path length reduction by at least 0.07%. This provides a compelling solution for effective robot path planning.

Although the proposed enhanced A-star algorithm has achieved significant success in robot path planning, we acknowledge that there is still room for improvement, warranting further in-depth exploration. Firstly, in scenarios with multiple obstacles between the start and end points, the algorithm's runtime may increase, offering a potential direction for further optimization. Future research will focus on refining the algorithm to make it more adaptable to different types of maps, addressing the challenges of complex environments. Secondly, we plan to extend the enhanced A-star algorithm to multi-robot collaborative path planning, aiming to enhance the overall system efficiency and coordination. Research in these two directions will contribute to further improving the algorithm's performance, making it more practical and versatile.

Acknowledgments

This work is partly supported by the National Natural Science Foundation of China (62271164, 62101158), the Major Scientific and technological innovation project of Shandong Province of China (2020CXGC010705, 2021ZLGX-05 and 2022ZLGX04), Shandong Provincial Natural Science Foundation (ZR2020MF017, ZR2022MF255, ZR2023MF051), the engineering research centre of Shandong province, The joint innovation centre of Shandong province. Shandong Provincial Key Laboratory of Marine Electronic Information and Intelligent Unmanned Systems, Key Laboratory of Cross-Domain Synergy and Comprehensive Support for

Unmanned Marine Systems, Ministry of Industry and Information Technology, Discipline construction fund (2023SYLHY05).

References

[1] L. Xu, X. B. Cao, W. B. Du, Y. M. Li, Cooperative path planning optimization for multiple UAVs with communication constraints, *Knowledge-Based Systems*, Vol. 260, Article No. 110164, January, 2023.

[2] E. Malayjerdi, R. Sell, M. Malayjerdi, A. Udal, M. Bellone, Practical path planning techniques in overtaking for autonomous shuttles, *Journal of Field Robotics*, Vol. 39, No. 4, pp. 410-425, June, 2022.

[3] T. T. Sang, J. C. Xiao, J. F. Xiong, H. Y. Xia, Z. Z. Wang, Path Planning Method of Unmanned Surface Vehicles Formation Based on Improved A* Algorithm, *Journal of Marine Science and Engineering*, Vol. 11, No. 1, Article No. 176, January, 2023.

[4] G. Tang, C. Q. Tang, C. Claramunt, X. Hu, P. Zhou, Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment, *IEEE Access*, Vol. 9, pp. 59196-59210, March, 2021.

[5] X. F. Yang, Y. L. Shi, W. Liu, H. Ye, W. B. Zhong, Z. G. Xiang, Global path planning algorithm based on double DQN for multi-tasks amphibious unmanned surface vehicle, *Ocean Engineering*, Vol. 266, Article No. 112809, December, 2022.

[6] Z. H. Liu, H. B. Liu, Z. G. Lu, Q. L. Zeng, A dynamic fusion pathfinding algorithm using Delaunay triangulation and improved A-star for mobile robots, *IEEE Access*, Vol. 9, pp. 20602-20621, January, 2021.

[7] G. Chen, T. Wu, Z. Zhou, Research on ship meteorological route based on A-star algorithm, *Mathematical Problems in Engineering*, Vol. 2021, pp. 1-8, May, 2021.

[8] S. Sedighi, D. Nguyen, K. Kuhnert, Guided hybrid A-star path planning algorithm for valet parking applications, *2019 5th international conference on control, automation and robotics (ICCAR)*, Beijing, China, 2019, pp. 570-575.

[9] Q. Gu, F. Q. Dou, F. Ma, Energy optimal path planning of electric vehicle based on improved A* algorithm, *Transactions of the Chinese Society for Agricultural Machinery*, Vol. 46, No. 12, pp. 316-322, December, 2015.

[10] E. S. Ueland, R. Skjetne, A. R. Dahl, Marine autonomous exploration using a lidar and slam, *International Conference on Offshore Mechanics and Arctic Engineering*, Trondheim, Norway, 2017, Article No. V006T05A029.

[11] A. Stentz, Optimal and efficient path planning for partially-known environments, *Proceedings of the 1994 IEEE international conference on robotics and automation*, San Diego, CA, USA, 1994, pp. 3310-3317.

[12] W. Y. Yue, J. Franco, W. Cao, H. W. Y, ID* Lite: improved D* Lite algorithm, *Proceedings of the 2011 ACM Symposium on Applied Computing*, TaiChung, Taiwan, 2011, pp. 1364-1369.

[13] C. Huang, H. Huang, J. Z. Zhang, P. Hang, Z. X. Hu, C. Lv, Human-machine cooperative trajectory planning and tracking for safe automated driving, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 8, pp. 12050-12063, August, 2022.

[14] J. C. Zhang, Y. Q. An, J. N. Cao, S. Ouyang, L. Wang, UAV Trajectory Planning for Complex Open Storage Environments Based on an Improved RRT Algorithm, *IEEE Access*, Vol. 11, pp. 23189-23204, March, 2023.

[15] S. C. Su, X. Ju, C. J. Xu, Y. F. Dai, Collaborative Motion Planning Based on the Improved Ant Colony Algorithm for Multiple Autonomous Vehicles, *IEEE Transactions on Intelligent Transportation Systems*, pp. 1-11, March, 2023. DOI: 10.1109/TITS.2023.3250756

[16] Z. Wang, X. Feng, H. D. Qin, H. M. Guo, G. J. Han, An AUV-Aided Routing Protocol Based on Dynamic Gateway Nodes for Underwater Wireless Sensor Networks, *Journal of Internet Technology*, Vol. 18, No. 2, pp. 333-343, March, 2017.

[17] X. J. Liu, Q. Gu, C. L. Yang, Path planning of multi-cruise missile based on particle swarm optimization, *2019 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, Beijing, China, 2019, pp. 910-912.

[18] W. L. Hao, C. Wu, S. C. Lin, Research on UAV path planning based on improved particle swarm algorithm with inertia weight, *2023 IEEE International Conference on Control, Electronics and Computer Technology (ICCECT)*, Jilin China, 2023, pp. 738-741.

[19] J. P. Tu, S. X. Yang, Genetic algorithm based path planning for a mobile robot, *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, Taipei, Taiwan, 2003, pp. 1221-1226.

[20] F. Wang, Z. W. Wang, M. J. Lin, Robot Path Planning Based on Improved Particle Swarm Optimization, *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, Nanchang, China, 2021, pp. 887-891.

[21] J. Y. Liu, X. X. Wei, H. J. Huang, An improved grey wolf optimization algorithm and its application in path planning, *IEEE Access*, Vol. 9, pp. 121944-121956, August, 2021.

[22] M. Maaz, A. Shaker, H. Cholakkal, S. Khan, S. W. Zamir, R. M. Anwer, F. S. Khan, Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications, *European Conference on Computer Vision*, Tel Aviv, Israel, 2022, pp. 3-20.

[23] Y. L. Wei, W. Y. Jin, Intelligent vehicle path planning based on neural network Q-learning algorithm, *Fire Control & Command Control*, Vol. 44, No. 2, pp. 46-49, February, 2019.

[24] Y. X. Long, H. J. He, Robot path planning based on deep reinforcement learning, *2020 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, Shenyang, China, 2020, pp. 151-154.

[25] U. Orozco-Rosas, K. Picos, O. Montiel, Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots, *IEEE Access*, Vol. 7, pp. 156787-156803, October, 2019.

[26] L. Ye, F. Y. Wu, X. J. Zou, J. Li, Path planning for mobile robots in unstructured orchard environments: An improved kinematically constrained bi-directional RRT approach, *Computers and Electronics in Agriculture*, Vol. 215, Article No. 108453, December, 2023.

[27] U. Orozco-Rosas, O. Montiel, R. Sepúlveda, Mobile robot path planning using membrane evolutionary artificial potential field, *Applied Soft Computing*, Vol. 77, pp. 236-251, April, 2019.

[28] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, A. Cuesta-Infante, Mobile robot path planning using a QAPF learning algorithm for known and unknown environments, *IEEE Access*, Vol. 10, pp. 84648-84663, August, 2022.

[29] Y. Ma, Y. J. Zhao, Z. X. Li, X. P. Yan, H. X. Bi, G. Królczyk, A new coverage path planning algorithm for unmanned surface mapping vehicle based on A-star based searching, *Applied Ocean Research*, Vol. 123, Article No.

103163, June, 2022.
[30] J. Zhang, J. Wu, X. Shen, Y. S. Li, Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star, *International Journal of Advanced Robotic Systems*, Vol. 18, No. 5, Article No. 17298814211042730, September-October, 2021.
[31] C. B. Wang, L. Wang, J. Qin, Z. Z. Wu, L. Duan, Z. Q. Li, M. Q. Cao, X. C. Ou, X. Su, W. G. Li, Z. Lu, M. Li, Y. Wang, J. Long, M. Huang, Y. Li, Q. Wang, Path planning of automated guided vehicles based on improved A-Star algorithm, *2015 IEEE International Conference on Information and Automation*, Lijiang, China, 2015, pp. 2071-2076.
[32] J. Lin, Improved A* Algorithm for Intelligent Warehouse Logistics Robot Path Planning, *Journal of Sanming University*, Vol. 38, No. 6, pp. 51-58, December, 2021.
[33] Y. Y. Guo, J. Yuan, K. G. Zhao, Robot path planning based on an improved A* algorithm and an improved dynamic window method, *Computer Science and Engineering*, Vol. 44, No. 7, pp. 1273-1281, July, 2022.
[34] X. W. Wang, J. J. Lu, F. Y. Ke, X. Wang, W. Wang, Research on AGV task path planning based on improved A* algorithm, *Virtual Reality & Intelligent Hardware*, Vol. 5, No. 3, pp. 249-265, June, 2023.
[35] W. G. Li, X. Su, AGV path planning based on improved A* algorithm, *Modern Manufacturing Engineering*, No. 10, pp. 33-36, October, 2015.
[36] E. Shang, B. Dai, Y. M. Nie, Q. Zhu, L. Xiao, D. W. Zhao, A guide-line and key-point based A-star path planning algorithm for autonomous land vehicles, *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, Rhodes, Greece, 2020, pp.1-7.
[37] J. W. Yu, J. Hou, G. Chen, Improved safety-first A-star algorithm for autonomous vehicles, *2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM)*, Shenzhen, China, 2020, pp. 706-710.
[38] E. Shang, B. Dai, Y. M. Nie, Q. Zhu, L. Xiao, D. W. Zhao, An improved A-Star based path planning algorithm for autonomous land vehicles, *International Journal of Advanced Robotic Systems*, Vol. 17, No. 5, Article No. 1729881420962263, September-October, 2020.
[39] T. Zheng, Y. Q. Xu, D. Zheng, AGV path planning based on improved A-star algorithm, *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Chongqing, China, 2019, pp. 1534-1538.
[40] X. H. Wang, X. H. Liu, Y. C. Wang, A Research on Task Scheduling and Path Planning of Mobile Robot in Warehouse Logistics Based on Improved A* algorithm, *Industrial Engineering Journal*, Vol. 22, No. 6, pp. 34-39, December, 2019.
[41] C. J. Huang, C. H. Wu, Application of improved A* algorithm in intelligent AGV path planning, *Journal of Putian University*, Vol. 22, No. 5, pp. 36-39, October, 2015.
[42] O. A. R. A. Wahhab, A. S. Al-Araji, An Optimal Path Planning Algorithms for a Mobile Robot, *Iraqi Journal of Computers, Communications, Control & Systems Engineering (IJCCCE)*, Vol. 21, No. 2, pp. 44-58, June, 2021.
[43] O. A. R. A. Wahhab, A. S. Al-Araji, Path Planning and Control Strategy Design for Mobile Robot Based on Hybrid Swarm Optimization Algorithm, *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 3, pp. 565-579, June, 2021.
[44] J. F. Lian, W. T. Yu, K. Xiao, W. R. Liu, Cubic spline interpolation-based robot path planning using a chaotic adaptive particle swarm optimization algorithm, *Mathematical problems in engineering*, Vol. 2020, pp. 1-20, February, 2020.
[45] C. W. Miao, G. Z. Chen, C. L. Yan, Y. Y. Wu, Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm, *Computers & Industrial Engineering*, Vol. 156, Article No. 107230, June, 2021.

Biographies

Xiao-Zhen Yan received her bachelor's, master's and doctor's degrees in Communication and Information Systems from Harbin Engineering University in 2005,2008 and 2012. Since 2021, she has been an associate professor at the School of Information Science and Engineering at Harbin Institute of Technology (Weihai). His main research interests include wireless sensor network, positioning and navigation.

Xin-Yue Zhou graduated from the University of Jinan in 2021 with a bachelor's degree. He is currently studying for a master's degree at Harbin Institute of Technology in Weihai. His main research direction is the multi-unmanned ship path planning technology.

Ruo-Chen Ding received her bachelor's and master's degrees from Harbin Institute of Technology (Weihai) in 2020 and 2022. His main research direction is intelligent storage path planning.

Qing-Hua Luo received his doctorate degree in Communication and Information Systems from Harbin Engineering University in 2005 and 2008, and his doctorate degree in Instrument Science and Technology from Harbin Institute of Technology in 2013. And has served as a professor since 2023. His main research interests include wireless sensor networks, uncertain data processing and fault diagnosis.

Chun-Yu Ju works as a Planning and Control Algorithm Engineer at Siasun Robotics and Automation Co., Ltd. He obtained a Master's degree in Control Science and Engineering from Harbin Institute of Technology in 2021. His research interests include path planning for autonomous driving and mobile robots.