Mathematical Biosciences and Engineering

MBE, 20(1): 145–178.
DOI: 10.3934/mbe.2023008
Received: 01 June 2022
Revised: 10 August 2022
Accepted: 25 August 2022
Published: 30 September 2022

http://www.aimspress.com/journal/MBE

Research article

Path planning and collision avoidance methods for distributed multi-robot systems in complex dynamic environments

Zhen Yang, Junli Li*, Liwei Yang*, Qian Wang, Ping Li and Guofeng Xia

School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650093, China

* **Correspondance:** Email: li_junli@stu.kust.edu.cn, 18916336783ylw@gmail.com.

Abstract: Multi-robot systems are experiencing increasing popularity in joint rescue, intelligent transportation, and other fields. However, path planning and navigation obstacle avoidance among multiple robots, as well as dynamic environments, raise significant challenges. We propose a distributed multi-mobile robot navigation and obstacle avoidance method in unknown environments. First, we propose a bidirectional alternating jump point search A* algorithm (BAJPSA*) to obtain the robot's global path in the prior environment and further improve the heuristic function to enhance efficiency. We construct a robot kinematic model based on the dynamic window approach (DWA), present an adaptive navigation strategy, and introduce a new path tracking evaluation function that improves path tracking accuracy and optimality. To strengthen the security of obstacle avoidance, we modify the decision rules and obstacle avoidance rules of the single robot and further improve the decision avoidance capability of multi-robot systems. Moreover, the mainstream prioritization method is used to coordinate the local dynamic path planning of our multi-robot systems to resolve collision conflicts, reducing the difficulty of obstacle avoidance and simplifying the algorithm. Experimental results show that this distributed multi-mobile robot motion planning method can provide better navigation and obstacle avoidance strategies in complex dynamic environments, which provides a technical reference in practical situations.

Keywords: distributed multi-mobile robots; path planning; A* algorithm; dynamic window approach; prioritization method

1. Introduction

The technology of autonomous mobile robots is developing rapidly, and it is widely used in entertainment, mining industry, education, medical services, military reconnaissance, agricultural automation, planetary exploration, and other fields [1]. Path planning and obstacle avoidance technology are critical to achieve autonomous robot navigation, which determines the application prospects of mobile robots. Meanwhile, multi-robot obstacle avoidance technology remains a relevant research problem in dynamic and complex environments.

Path planning is generally divided into global and local path planning. Global path planning involves planning an optimal or suboptimal safe path with priori map information [2]. In contrast, local path planning is designed with dynamic obstacle environments in real-time. The robot usually must acquire details about the environment, including the coordinates of static and dynamic obstacles, with the help of a local path planner [3].

Current global path planning in a known environment has attracted significant interest. Numerous algorithms have been explored, including the A* algorithm [4–6], ant colony optimization [7,8], particle swarm optimization [9,10], bacterial foraging optimization [11,12], bat algorithm [13,14], and whale optimization algorithm [15,16], etc. With the advantages of a simple structure, facilitated implementation, and fast planning, the A* algorithm has been popular among researchers [17]. Wang et al. [5] used a bidirectional search strategy to improve the A* algorithm, which significantly enhanced the search performance by simultaneously conducting the iterative search in both positive and negative directions. Zhang et al. [6] enhanced the node expansion method of the A* algorithm based on the jump point search (JPS) strategy, which significantly reduced the memory overhead and the search scale. We further mention some research on intelligent optimization algorithms. Miao et al. [8] introduced an angle guidance factor and an obstacle exclusion factor in the transfer probability of ant colony optimization, and the global search ability and convergence speed of the algorithm were balanced. Song et al. [9] combined an adaptive fractional-order, velocity-improved PSO algorithm with the continuous high-degree Bezier curve to plan smooth paths for mobile robots. Hossain et al. [11] searched for the shortest path in a dynamic environment based on the bacterial foraging optimization algorithm. Tang et al. [13] presented the first application of the bat algorithm to a collaborative multi-robot search task in an unknown environment. They used adaptive inertial weights and the Doppler effect to improve the frequency formulation to avoid premature convergence. Yan et al. [15] proposed a whale optimization algorithm based on the forward-looking sonar to solve the 3D path planning problem for UUVs, with strong stability and search capability.

The studies mentioned above [4–16] conducted some work to improve the efficiency of path planning. However, they yielded a few practical solutions to the obstacle avoidance problem of mobile robots in the actual dynamic environments. Ensuring the safety of robots with the help of local path planning is an effective solution when the environment is dynamic and full of uncertainties [18], and the main popular local path planning algorithms are the dynamic window approach (DWA) [19] and artificial potential field method (APF) [20], etc. DWA is a highly efficient, real-time obstacle avoidance algorithm that transforms the path planning problem into the constrained optimization problem of the velocity space and controls the robot motion by outputting the optimal real-time speed [19]. However, DWA faces problems, such as local optima and a low successful obstacle avoidance rate for dynamic obstacles. Therefore, Chang et al. [21] modified and extended the evaluation function of DWA and used the reinforcement learning method to adaptively adjust parameters, which further enhanced the planning effect. Lin et al. [22] improved the avoidance rate of DWA against dynamic obstacles by using a fuzzy control scheme to evaluate the danger level of moving obstacles through collision risk index and relative distance. Furthermore, there has been significant interest in APF. Cheng et al. [23] introduced optimal control theory to reformulate the UAV path planning problem as a constrained optimization problem with APF. Orozco et al. [24] solved path planning problems in dynamic environments by combining membrane computing with a genetic algorithm and APF. In general, it is popular to combine local path planning algorithms with global ones to cope with environments of increasing complexity and uncertainty. The hybrid algorithms allow the robot to connect global path optimality and stochastic obstacle avoidance to a relatively large extent [25]. Ji et al. [26] combined the A* algorithm with an adaptive DWA for global path planning research that solves robot motion in a complex environment. Wang et al. [27] combined the improved PSO algorithm with APF for USVs to solve the dynamic path planning problem in complex offshore regions.

The studies mentioned above [19–27] explored the global or local path problems from different perspectives, but with less attention to multi-robot obstacle avoidance. The aim of multi-robot path planning is to find a conflict-free path from the start to the target for each robot. The motion of mobile robots is disturbed not only by known factors in the global environment, but also by dynamic obstacles and other autonomous robots, which making it necessary and practical to design an obstacle avoidance system for multiple autonomous robots. In the context of research on multi-robot strategies, the main approaches are either centralized or distributed. The centralized approach considers the cost or objective function, where the constraints for all robots are considered together, thus obtaining the paths of individual robots in a global search. It prioritizes completeness with less attention to the personal robot [28]. One of the more popular ways to employ the centralized approach is the formation control, where the mission planning information and formation information is integrated into a leader robot, while the other robots act as followers. The leader coordinates the actions of each follower to maintain the formation from the start to the end. Dai et al. [29] proposed a multi-robot formation switching strategy incorporating a priority model, where the leader robot with the highest priority is responsible for planning a safe path and guiding the follower robots, and the following robots switch into an obstacle avoidance formation by calculating the desired distance and angle. Sang et al. [30] combined A* and APF for the USVs formation problem, using the A* algorithm to plan the globally optimal path, dividing it into multiple sub-target points, and used the improved APF for path tracking and performing formation obstacle avoidance. In distributed multi-robot path motion planning, each robot independently determined its collision-free trajectory path towards the goal without colliding with static obstacles or colleagues. The navigation problem for a distributed-based multi-robot is divided into path planning and movement phases, planning a globally optimal path for each robot and maintaining the safety of multi-robot movement. Das et al. [31] added the consideration of path deviation and energy consumption optimization by embedding the social and cognitive behavior of an improved particle swarm algorithm (IPSO) into the Newtonian gravity of an enhanced gravity search algorithm (IGSA). They proposed IPSO-IGSA to implement path planning for multiple robots in dynamic environments and improve search capability by simultaneously updating particle positions using IPSO velocity and IGSA acceleration. In subsequent research, the authors [31] further investigated the multi-robot collision-free planning problem by mixing improved particle swarm optimization (IPSO) and evolutionary operators (EOPs)

[32]. Further, some scholars [33] set different priorities for each robot by the prioritization method, thus reducing the possibility of robot collisions.

This study proposes a distributed multi-robot navigation and obstacle avoidance method in unknown environments, applying it to path planning and navigation. The main contributions are as follows:

1) In global path planning:

A jump point search strategy and a bidirectional alternating search strategy are introduced to the conventional A* algorithm, and heuristic functions are designed based on its characteristics, called BAJPSA*, which we efficiently obtain the robots' globally optimal path.

2) In local path planning:

- First, considering dynamics and environmental constraints, the robots are constructed based on DWA. Then, the adaptive navigation strategy and path deviation evaluation function are proposed for improving the path tracking accuracy and optimality of our robots.
- Second, according to the potential collision situations between the robot and dynamic obstacles, we improved the obstacle recognition method and designed three obstacle avoidance rules, which increase the robot's success rate in avoiding dynamic obstacles with a higher move velocity or bigger size.
- Finally, the distributed multi-robot systems are extended from our above single-robot obstacle avoidance algorithm. Focusing on the motion conflicts among multiple robots, we propose a collision recognition strategy and fuse it with a task prioritization strategy to coordinate the robots' motion and obstacle avoidance.

This paper is organized as follows: Section 2 describes our BAJPSA* algorithm. Section 3 describes our multi-robot motion planning algorithm. Section 4 discusses the experiments. Section 5 concludes the whole paper and discusses future work.

2. Global path planning based on BAJPSA* algorithm

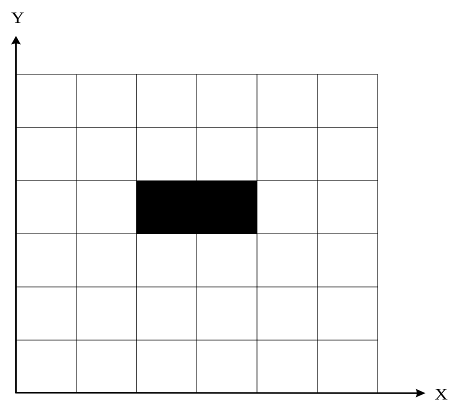
2.1. Conventional A algorithm*

The A* algorithm is a classical heuristic search algorithm, where the algorithm selects the node with the smallest evaluation value as the next expanded node in the search process [34], and the evaluation function is expressed as:

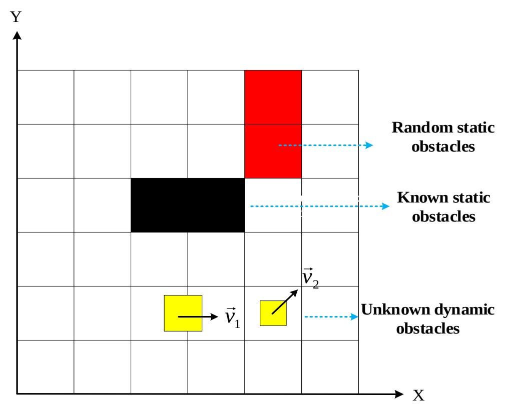
$$f(n) = g(n) + h(n) \tag{1}$$

where n is the current node and the evaluation function $f(n)$ is used to calculate the total cost of the current node; $g(n)$ is the actual cost from the starting point to the current node n; $h(n)$ is the heuristic function used to estimate the cost from the current node n to the target point.

The A* algorithm is a search method based on grid traversal [35], such that it must establish a suitable motion environment for mobile robots. A 2D grid map is shown in Figure 1, where Figure 1(a) is the most widely investigated environment for robots with a priori knowledge, containing black obstacle regions and passable white regions. Figure 1(b) represents our multi-robot motion environment considering practical factors, expanding the unknown local elements of the robot, further including unknown static obstacles (red grid) and unknown volume size and motion velocity of the obstacles (yellow grid).



(a) Traditional global static environment



(b) Global static + local dynamic environment in this paper

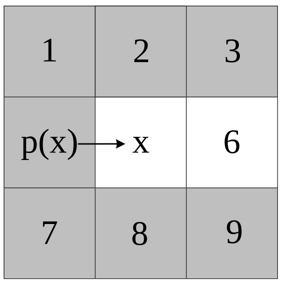
Figure 1. Grid environment.

2.2. Conventional jump point search (JPS) algorithm

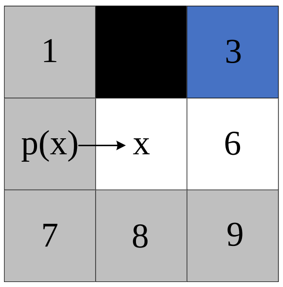
The JPS algorithm was proposed by Daniel Harabor and Alban Grastien [36,37] in 2011, which is based on the A* algorithm to find paths by defining and computing heuristic values for those nodes on the uniform cost grid graph where the jumping rules are satisfied. It is several orders of magnitude faster than the A* algorithm in terms of computational speed, and the memory overhead as well as the computational effort are significantly reduced, which has been proved by Harabor et al. [36]. The main steps of JPS include two parts [36]: (1) pruning rules, which filter out the nodes in the grid map that do not need to be expanded and eliminate them. (2) Jumping rules, which identify the jump nodes in the grid map and evaluate them.

2.2.1. Pruning rules

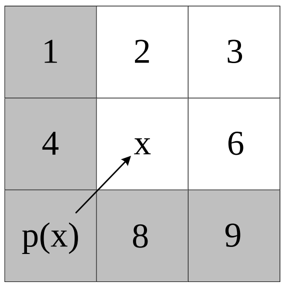
The set of neighboring nodes around node x is defined as the $neighbors(x)$, and the cost of moving one grid in straight is 1, and diagonal is $\sqrt{2}$. The function of pruning rules are to recursively prune the set of neighbours around each node, which means pruning all nodes that can be reached optimally by a path that does not visit the current node. Besides, the process of pruning rules is performed entirely online, involves no preprocessing and has no memory overhead.



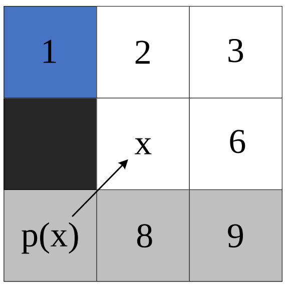
(a) straight direction



(b) forced neighbor node



(c) diagonal direction



(d) forced neighbor node

Figure 2. Pruning rules.

Situation 1: $neighbors(x)$ contains no obstacle

1) Straight moves

When node x is not adjacent to an obstacle, and the algorithm is extended along the straight direction, the node $n \in neighbors(x)$ that satisfies Eq (2) will be pruned.

$$len(\langle p(x), \dots, n\rangle \backslash x) \leq len(\langle p(x), x, n\rangle) \tag{2}$$

where len represents the cost of the path; $\langle p(x), x, n\rangle$ represents the path from parent node $p(x)$ to node n through node x; $\langle p(x), \dots, n\rangle \backslash x$ represents the path from parent node $p(x)$ to node n directly without passing through node x.

As shown in Figure 2(a), there exists a path $\pi' = \langle p(x), 2\rangle$ from node $p(x)$ to node 2 that is shorter than the path $\pi = \langle p(x), x, 2\rangle$ from node $p(x)$ to node 2 via node x. Therefore, the gray nodes{1, 2, 3, 7, 8, 9} need to be pruned according to Eq (2). Only the remaining node 6, marked white, needs to be considered, which is called the natural neighbor of node x.

2) Diagonal moves

When node x is not adjacent to an obstacle, and the algorithm is extended along the diagonal direction, the node $n \in neighbors(x)$ that satisfies Eq (4) will be pruned.

$$len(\langle p(x), \dots, n\rangle \backslash x) < len(\langle p(x), x, n\rangle) \tag{3}$$

As shown in Figure 2(c), there exists a path $\pi' = \langle p(x), 4{,}1\rangle$ that is shorter than the path $\pi = \langle p(x), x, 1\rangle$ that goes through node x. All gray nodes n that satisfy this condition, such as 1, 4, 8 and 9 will be pruned, and the leaved white node 2, 3 and 6 are the natural neighbors of the current node x.

Situation 2: $neighbors(x)$ contains an obstacle

Furthermore, when $neighbors(x)$ contains an obstacle, Eq (2) will not be able to prune all non-natural neighbors due to the presence of obstacles. Therefore, the concept of the forced neighbor is introduced. A node $n \in neighbors(x)$ is forced if:

1) n is not a natural neighbor of node x;

2) n satisfies the rule of Eq (3).

$$len(\langle p(x), x, n\rangle) < len(\langle p(x), \dots, n\rangle \backslash x) \tag{4}$$

As shown in Figure 2(b), node 3 is a non-natural neighbor of node x, and the path $\pi' = \langle p(x), 8{,}6{,}3\rangle$ is longer than the path $\pi = \langle p(x), x, 3\rangle$ from node $p(x)$ to node 3 via node x. Therefore, node 3 marked with blue need to be forcedly considered. Similar to Figure 2(b), node 1 in Figure 2(d) is also a forced neighbor of node x

2.2.2. Jumping rules

Node y is the jump point from node x , heading in direction $\overrightarrow{d}$, if y minimizes the value k such that the $y = x + k\overrightarrow{d}$ and one of the following conditions holds:

(1) Node y is the target node.

(2) Node y has at least one neighbour that is a forced neighbor.

(3) There exists a node $z = y + k_i\overrightarrow{d_i}$ that lies $k_i \in N$ steps in direction $\overrightarrow{d_i}$ and node z is a jump point successor of node y according to condition (1) or condition (2).

where $\overrightarrow{d}$ represents a diagonal move, and $\overrightarrow{d_i}$ represents two straight moves at 45° to $\overrightarrow{d}$ as $\overrightarrow{d_1}$ and $\overrightarrow{d_2}$. $y = x + k\overrightarrow{d}$ represents that node y can be reached by taking k unit moves from node x in diagonal direction $\overrightarrow{d}$. $z = y + k_i\overrightarrow{d_i}$ represents that node z can be reached by taking k_i unit moves from node y in straight direction $\overrightarrow{d_i}$.

Figure 3 shows an example of a jump point identified by Condition 3. The dashed line indicates the process of the JPS algorithm searching along the diagonal direction after failing in a straight direction, and the solid lines indicate the path formed by node x and jump points. According to Condition 2 of the jumping rules, nodes x and z have a forced neighbor w and v, respectively, so nodes x and z are jump points. According to Condition 3, node y can be reached along the horizontal direction from jump point z, such that node y is also a jump point.

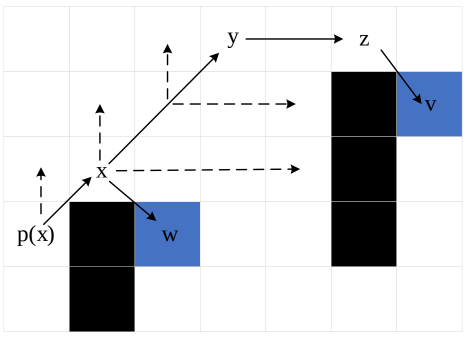


Figure 3. Example of jumping rules.

2.3. Bidirectional alternating search strategy

Bidirectional search defines the forward search from the starting point to the target point and the reverse search from the target point to the starting point, however, it has the following two problems:

1) As shown in Figure 4, the bidirectional search is conducted from the starting point and the target point at the same time, which may result in two different paths being searched.
2) Theoretically, forward and backward searching simultaneously search toward the target and starting points and meet at their geometric center [38]. In this case, the algorithm has the highest search efficiency. However, the obstacle density and distance between jump points are different, and the paths may not meet at the midpoint.

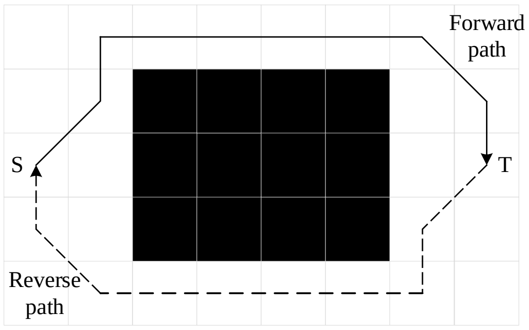


Figure 4. Pathfinding failure.

For the above reasons, we use a bidirectional alternating search strategy, where the forward and backward searches are alternated, and only the forward search finds the jump point before starting the backward search. In this way, the forward and reverse searches meet at the midpoint as much as possible and will benefit the search efficiency of our algorithm. The specific steps of the BAJPS strategy are as follows:

Step 1: Create two OPEN lists and two CLOSE lists: OPEN_1 and CLOSE_1 are used to store the jump points to be checked, and the expanded jump points in the forward expansion process, respectively, and OPEN_2 and CLOSE_2 are used to store the jump points to be checked and the expanded jump points in the reverse expansion process. Add the starting node S to OPEN_1, add the target node T to OPEN_2, and set both CLOSE lists to empty.

Step 2: Alternate forward and reverse iterative jump point searches, starting with the forward search.

(1) If there was at least one node in the list of OPEN_1, select the lowest cost node n based on the valuation function $f(n)$; if node n was the target point, the search process is terminated, and the path returned; otherwise, the node n is removed from the list of OPEN_1 and added to the list of CLOSE_1.

(2) Starting from node n, continue to search jump point p_{n1} in the direction of its natural successors. Horizontal and vertical search directions are executed preferentially and only consider diagonal directions when obstacles are encountered, or map boundaries are reached.

A. If there was no searched jump point or the returned node p_{n1} was in CLOSE_1, it is ignored.

B. If the returned node p_{n1} was not in the OPEN_1, add it to OPEN_1 and calculate its $g(n)$, $h(n)$, and $f(n)$. Regard the node n as the parent node of the node p_{n1}.

C. If the node p_{n1} was in the OPEN_1, update $g(n)$ and calculate whether $g(n)$ is below its previous value. If yes, change the node n as the parent node of the node p_{n1}. and calculate $f(n)$.

Step 3: The reverse search for jump point p_{n2}, with its corresponding parent node n_2, begins as soon as the forward search is completed and obtains jump point p_{n2}.

Step 4: The forward and backward jump points are searched alternately, and when there are the same jump points in the CLOSE list, the search would be finished.

Step 5: From the same jump point that appeared in Step 4, connect the jump points deposited in forward and reverse directions in sequence to obtain the eventual route.

2.4. Improving the heuristic function

The traditional A* algorithm uses the Euclidean distance, Manhattan distance, or Chebyshev distance to calculate the heuristic function [39] and the distance functions are as follows:

$$h(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{5}$$

$$h(n) = |x_2 - x_1| + |y_2 - y_1| \tag{6}$$

$$h(n) = max(|x_2 - x_1|, |y_2 - y_1|) \tag{7}$$

where (x_1, y_1) and (x_2, y_2) denote the coordinates of the current and the target nodes, respectively.

The A* algorithm with Manhattan distance performs a four-directional search. In contrast, considering that the Euclidean distance is expanded to a broader eight-neighborhood, the obstructive effect of obstacles within the environment will lead to the heuristic value of the evaluation function being smaller than the actual value. Therefore, we combine Euclidean distance and Chebyshev distance to design a heuristic function that is more consistent with our BAJPSA* algorithm as Eq (8).

The improved heuristic function can appropriately reduce the weight of the Chebyshev distance according to the JPS to improve the solution of the optimal path. The obtained heuristic value is closer to the actual path cost and further reduces the number of nodes to be evaluated, which improves the search efficiency of the algorithm.

$$h(n) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \frac{max(|x_2 - x_1|, |y_2 - y_1|)}{2} \tag{8}$$

The pseudo-code of path planning process based on the BAJPSA* algorithm is as follows:

Algorithm 1. Bidirectional Alternating Jump Point Search A* (BAJPSA*)

```
1:   Initialize the grid map;
2:   Open_1, Close_1; Open_2, Close_2 ← The jump points to be checked and the expanded jump points in
     the positive and reverse process, respectively;
     Put the starting point into Open_1; Put and the endpoint into Open_2;
3:   If Open_1 or Open_2 is an empty node;
4:     Pathfinding failed;
5:     Return;
6:   else
7:     While Positive node ~=Reverse node  do
8:       Calculate the f (n) value of all nodes in the Open_1 and Open_2 according to Equations (1) and
         (8) ;
9:       Close_1 and Close_2 ← The node with the smallest f (n) value in the Open_1 and Open_2,
         respectively;
10:      Positive node and Reverse node = the node with the smallest  f (n) value, respectively;
11:      If  Positive node ==Reverse node;
12:        The optimal path is obtained and the algorithm ends;
13:        Break;
14:      Else
15:        While (Positive node & Reverse node)~= [ ]  do
16:          Search for jump points horizontally and vertically alone the direction from the parent node to
             the current node;
17:          if Encounter obstacles or map edges;
18:            Search for jump points diagonally alone the direction from the parent node to the current
               node;
19:          Elseif  Search for jump points;
20:            Close_1 or Close_2 ← The searched jump points;
21:          Else
22:            Ignore the node, continue searching;
23:          end
24:        end
25:      end
26:    end
27:  end
```

3) Multi-robot motion planning

Research on mobile robots relying on multiple sensor fusion technologies to sense the surrounding environment information, combined with appropriate local path planning algorithms to avoid moving obstacles or seeking dynamic goals, has been among the most popular topics in the field of robotics in recent years [40]. The APF method is favored by scholars owing to its high flexibility and smooth planning trajectory [41]. However, there are problems such as path oscillation and difficulty in ensuring path optimality when facing the actual, more complex natural dynamic environment. This study proposes an improved dynamic window approach (DWA) in Section 3.4 with great real-time and flexibility to make our multi-robot adapt to more complex and changing environments.

3.1. Robot kinematic model

Considering the two-wheel differential robot kinematic model shown in Figure 5, $x(t), v(t)$ and $\theta(t)$ are the linear velocity, angular velocity and direction of motion of the robot at the current moment t, respectively. Then, the motion state of the robot at the moment $t+1$ can be expressed as:

$$\begin{cases} x(t+1) = x(t) + v(t)\Delta t cos(\theta(t)) \\ y(t+1) = y(t) + v(t)\Delta t sin(\theta(t)) \\ \theta(t+1) = \theta(t) + \omega(t)\Delta t \end{cases} \tag{9}$$

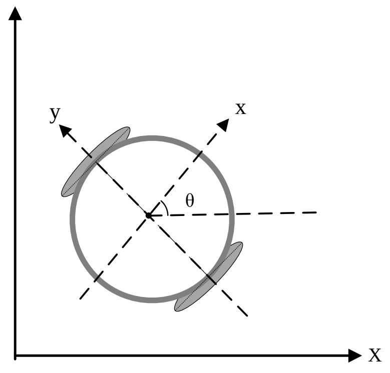
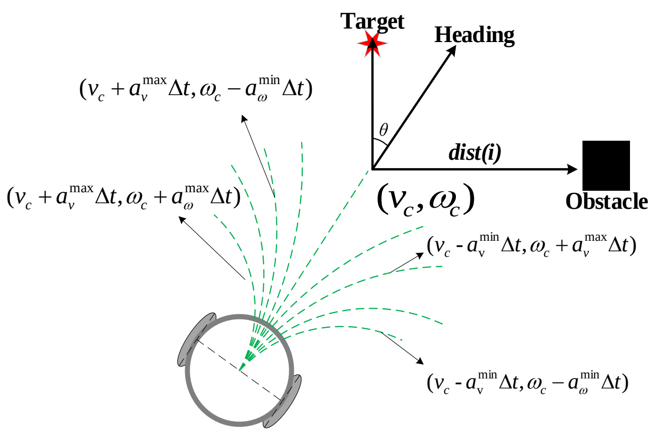


Figure 5. Kinematics model of wheeled robot.

3.2. Speed sampling

DWA describes the obstacle avoidance as an optimization problem with constraints in the velocity space. The conditions mainly include the incomplete constraints of the differential robot, the limitations of environmental obstacles, and the dynamics constraints of the robot structure. As shown in Figure 6, the search space of the robot is constrained by its maximum and minimum speed, motor performance, and braking distance to constrain the motion speed (v, w) within a certain range.



Target
Heading
θ
dist(i)
Obstacle
(v_c, ω_c)
$(v_c + a_v^{\max}\Delta t, \omega_c - a_\omega^{\min}\Delta t)$
$(v_c + a_v^{\max}\Delta t, \omega_c + a_\omega^{\max}\Delta t)$
$(v_c - a_v^{\min}\Delta t, \omega_c + a_v^{\max}\Delta t)$
$(v_c - a_v^{\min}\Delta t, \omega_c - a_\omega^{\min}\Delta t)$

Figure 6. Schematic diagram of robot constrained in velocity space.

According to the velocity limit of the robot, v_s is defined as the set of linear and angular velocities of the robot to reflect the maximum range of the search solution, and the velocity constraint of the robot is:

$$v_s = \{(v, \omega) | v \in [v_{min}, v_{max}], \omega \in [\omega_{min}, \omega_{max}]\} \tag{10}$$

In practice, the robot is limited by the motor torque constraints. It is theoretically impossible to reach the maximum and minimum reachable linear velocity v and angular velocity ω, such that the search range of the dynamic window is further reduced. Given the linear velocity v_c and angular velocity ω_c, the velocity v_d in the Δt sampling period under the considered motor constraint is:

$$v_d = \{(v, \omega) | v \in [v_c - a_v^{min}\Delta t, v_c + a_v^{max}\Delta t], \omega \in [\omega_c - a_\omega^{min}\Delta t, \omega_c + a_\omega^{max}\Delta t]\} \tag{11}$$

where v_c and ω_c are the linear and angular velocities at the current moment, respectively; a_v^{min} and a_ω^{min} are the minimum linear and the minimum angular deceleration, respectively; a_v^{max} and a_ω^{max} are the maximum linear and the maximum angular accelerations, respectively, and Δt is the sampling time.

The trajectory of the whole robot can be subdivided into several straight lines or circular arcs. To ensure the robot's safety area, the current speed must be able to decelerate to zero before hitting the obstacle under the maximum deceleration condition. Then, the braking distance of the robot is constrained as follows:

$$v_a = \{(v, \omega) | v \leq \sqrt{2 \cdot dist(v, \omega) a_v^{min}}, \omega \leq \sqrt{2 \cdot dist(v, \omega) a_\omega^{min}}\} \tag{12}$$

where $dist(v, w)$ is the distance between the simulated trajectory (based on velocity group (v, w)) and the nearest obstacle, that the simulated speed must satisfy $0 - v_a^2 = -2dist(v, w)a_v^{min}$and $0 - w_a^2 = -2dist(v, w)a_w^{min}$ to guarantee the robot's safety to a greater extent.

In summary, according to the three constraints of the robot search space, the input range for velocity control can be expressed as follows:

$$v_r = v_s \cap v_d \cap v_a \tag{13}$$

3.3. Evaluation function

The robot's linear velocity $v(t)$ and angular velocity $\omega(t)$ are sampled and combined with its kinematic model to simulate several trajectories within ns. The evaluation function selects the trajectory with the highest evaluation value, and the corresponding velocity group (v, w) is passed to the robot motion. The traditional evaluation function is as follows:

$$G(v,\omega) = \sigma(\alpha \cdot Head(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\omega)) \tag{14}$$

where $Head(v,\omega)$ is the navigation function, which indicates the azimuthal deviation between the end direction of the trajectory and the current target point; $dist(v,\omega)$ is the obstacle avoidance function, which shows the distance between the trajectory and the nearest obstacle; $vel(v,\omega)$ is the evaluation function of the robot motion speed at the current moment; σ is the normalization process; α, β and γ are the weighting coefficients of the corresponding evaluation functions, respectively.

3.4. Improved DWA

The widest method [26] takes the turning points of global path planning as the crucial waypoint to guide the robot's motion. However, this is not suitable for the case of power inspection robots, where the global path tracking accuracy must be strictly guaranteed. Besides, traditional DWA is ineffective for avoiding dynamic obstacles in an unknown environment and is highly susceptible to collision with such obstacles. To increase the obstacle avoidance and global path tracking capability of our multi-robot systems in a dynamic environment, we enhance the performance of the evaluation function of conventional DWA and propose a solution to the multiple conflicts that exist between dynamic obstacles and multiple robots.

3.4.1. Improving robot trajectory tracking capability

1) Improvement of $Head(v,\omega)$ evaluation function target point tracking method

In related studies [42], the most critical nodes that provide navigation information for robots are turning points of the global path, and the path tracking accuracy is poor. We investigated the method of Yang et al. [40] that extracted the nodes of three times B-spline paths as key navigation points, and designed the function $Remake[.]$ to reorganize the path Route generated by BAJPSA*, as shown in Figure 7 and Eq (15):

$$NewRoute = \sum_{i=2}^{m} Remake[Route(i), Route(i-1), n_{ds}] \tag{15}$$

where $Remake[.]$ is the crucial navigation point extraction function, designed in the following way: First, connect the adjacent path points $Route(i)$ and $Route(i-1)$, following form a line equation, then solve for the sequence of node coordinates (x, y) that satisfies the desired node distance n_{ds} on that line from the starting point $Route(i-1)$ and deposit them into $NewRoute$ in turn until all equations composed of the path $Route$ are cycled through; m is the number of critical nodes of $Route$ (includes: start point, endpoint and turning points).

To avoid continuous acceleration as well as deceleration and improve the accuracy of the robot's trajectory tracking, we set the desired distance d_1. When the distance $distance(.)$from the endpoint $tra(x, y)$of the optimal trajectory evaluated by Eq (14) to the target point $target(t)$ at moment t is less than the expected distance d_2, the critical navigation point at the moment $t+1$ is obtained from $NewRoute(.)$ in advance.

$$target(t+1) = NewRoute(floor(j \cdot d_1/n_{ds})), distance(tra, target(t)) < d_2 \quad (16)$$

where $target(t+1)$ represents the navigation information point of the robot at moment $t+1$, that is, when the distance between the optimal trajectory $tra(x,y)$ and the target point is less than d_2, the next navigation point changes from $NewRoute(floor(j(1) \cdot d_1/n_{ds}))$ to $NewRoute(floor(j(2) \cdot d_1/n_{ds}))$, j is a sequence of consecutive positive integers; $floor(.)$ is a rounding operation and the number of path node intervals can be estimated by the desired distance d_1 and the desired inter-node distance n_{ds}.

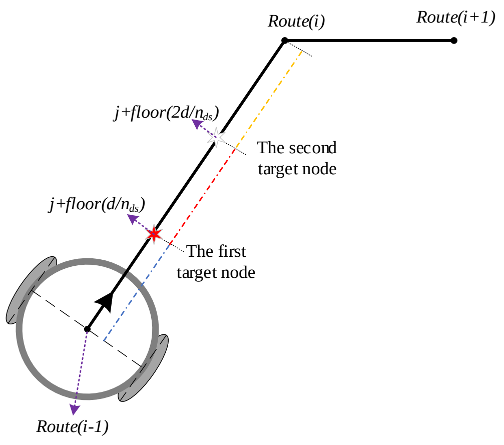
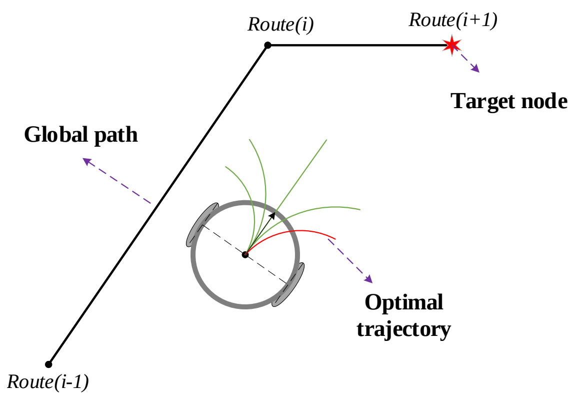


Figure 7. Schematic diagram of key target point extraction method.

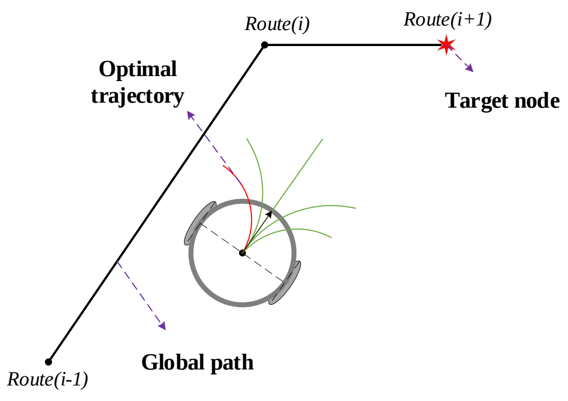
Take Figure 7 as an example, we can analyze the action of Eqs (15) and (16) in more detail: the original path contains three path nodes $Route\ (i-1,\ i,\ i+1)$, then the new path NewRoute with a large amount of node information is generated by Eq (15) and the distance between every two nodes is $\mathrm{n_{ds}}$; Next, we set the desired distance d_1 in Eq (16) as a way to extract the robot's motion navigation points from the new path $NewRoute$. Furthermore, when the condition $distance(tra, target(t)) < d_2$ is satisfied, the robot will receive the new navigation information in advance.

2) New $path(v,\omega)$ evaluation function

As shown in Figure 8, the experimental results in relevant literature indicate that most robots tend to deviate from the global path to some extent near the turning point, primarily due to the complexity of the environment. To make our robot consider the degree of global deviation during local path selection, we propose a new $path(v,\omega)$ function based on the original evaluation function to ensure that the robot moves along the global path as much as possible. The improved evaluation function as Eq (17) and the pseudo-code of the improved dynamic window approach is shown in Algorithm 2.



(**a**) Robotic pathfinding in some literature



(**b**) Robotic pathfinding in this study

Figure 8. Description of $path(v,\omega)$ evaluation function.

$$G(v,\omega) = \sigma(A \cdot Head(v,\omega) + B \cdot dist(v,\omega) + C \cdot vel(v,\omega) + D \cdot path(v,\omega)) \tag{17}$$

Algorithm 2. Improved dynamic window approach (IDWA)

```
1:    Initializing (grid map, Robot, Evaluation_Factor);
2:    Robot= [$v_{max}, \omega_{max}, a_v^{max}, a_\omega^{max}, Velocity\ Resolution, RPM\ Resolution$];
3:    Evaluation_Factor = [A; B; C; D];
4:    global_path ←Algorithm1 (BAJPSA* );
5:    sensor_messages ← Robot;
6:    if the trajectory without obstacles ← the Traditional DWA combined with the strategy in Section
      3.4.2 to filter trajectories
7:      while the local target location is not reached  do  ← the local target point information is
        obtained from equations (15-16)
8:        Speed sampling of robot;
9:        Simulate motion trajectories;
10:       Use the improved evaluation function (17) to select the optimal trajectory;
11:       Robot follows the optimal trajectory to move;
12:     end
13:   end
```

Considering the complex dynamic environment and environmental characteristics of multi-robot work comprehensively, our robot must solve not only the path fitting problem at the turning point, but also the path offset problem during dynamic obstacle avoidance. To this end, we design three $path(v,\omega)$ functions to correct the global path tracking capability of the robot by considering the distance relationship between the robot and the obstacles as well as the global path.

Situation 1: If the robot is far from the obstacle but deviates from the global path to a lesser extent, global path tracking is guaranteed as a priority.

$$Path(v,\omega) = \frac{1}{1 + min(path)}, min(dist(v,\omega)) \geq l_1 \& \, min(path) \leq l_2 \tag{18}$$

where $path = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, (x_1, y_1) and (x_2, y_2) denote the local path coordinates planned by the robot according to the kinematic model and the global path coordinates obtained by our BAJPSA* algorithm, respectively; $min(dist(v, \omega))$ denotes the closest distance from the end of the predicted trajectory to the edge of the obstacle; $min(path)$ denotes the most relative distance from the robot to the global path; l_1 denotes the desired obstacle avoidance distance of the robot from the obstacle in the case of small deviation from the global path; l_2 denotes the maximum error of the robot from the global path.

Situation 2: If the robot is close to the obstacle and deviates from the global path to a small extent, the weight D of the $Path(v, \omega)$ function is 0. Then, our robot's obstacle avoidance effectiveness is guaranteed preferentially in Eq (17).

$$\begin{cases} Path(v, \omega) = \dfrac{1}{1 + min(path)}, min(dist(v, \omega)) < l_1 \,\&\, min(path) \le l_2 \\ D = 0 \end{cases} \tag{19}$$

Situation 3: If the robot is far from the obstacle and deviates from the global path to a large extent, the robot is prompted to move closer to the global path by increasing the evaluation metric of $Path(v, \omega)$ in Eq (17).

$$\begin{cases} Path(v, \omega) = min(path) \\ D = 1 \end{cases}, min(dist(v, \omega)) \ge l_3 \,\&\, min(path) > l_2 \tag{20}$$

where l_3 denotes the desired distance of the robot from the obstacle in the case of a large deviation from the global path.

3.4.2. Improving the dynamic obstacle avoidance capability of the robot

1) Dynamic obstacle recognition area

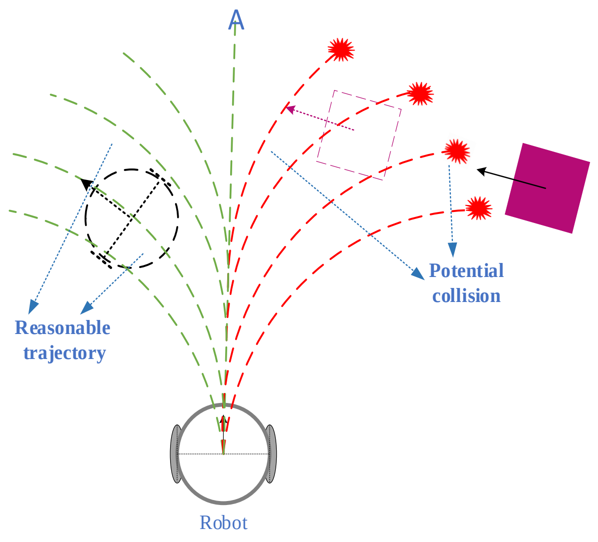


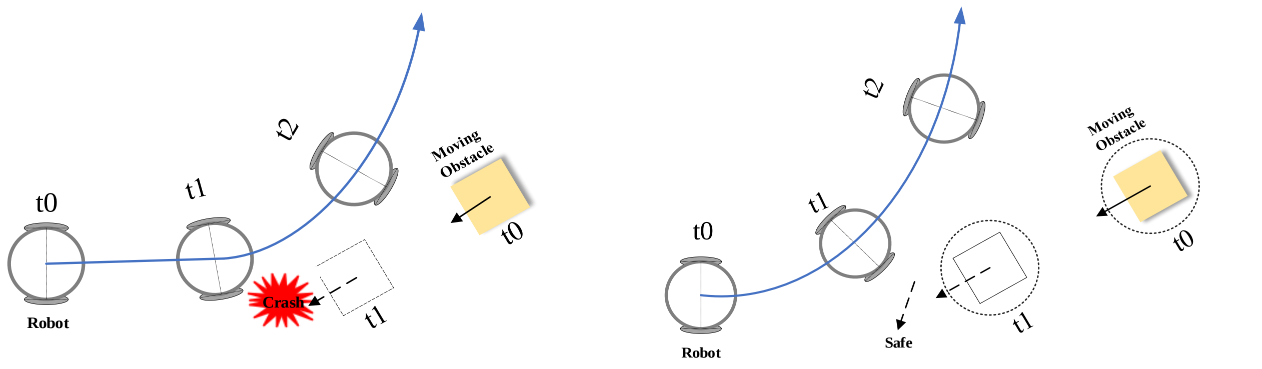
Figure 9. Robot dynamic obstacle avoidance search path schematic.

The traditional DWA does not identify whether the obstacle is dynamic or static when performing trajectory selection, such that the example shown in Figure 9 will mistakenly identify all red trajectories as collision trajectories and discard them. Based on the evaluation function, the robot most likely to select the path with the relatively best score from the green trajectories. However, dynamic obstacles (pedestrians, vehicles, etc.) are in constant motion, and the robot will continue to move in this way by selecting a green trajectory in the following path selection process. The final result is an awkward situation, where either robot collides with the obstacle or gets stuck in a local optimum of following the motion of the obstacle. A typical collision scenario is shown in Figure 10, where a conventional robot lacks effective recognition of dynamic obstacles to make timely decisions, and the collision occurs at the moment $t1$. In order to improve the safety and reliability of robot motion, our robot adds an appropriate recognition area for such moving obstacles, which reduces the risk of conflict to some extent.

In the natural environment, dynamic obstacles have different volume sizes, so we considered a circular recognition area that can accommodate the whole object. Considering grid environment effects and the movement speed of obstacles, the actual volume of dynamic obstacles in this paper are square-shaped and not more than one grid($1\ m$), and the robot's circle recognition radius R is as follows:

$$R = N \cdot r \ and \ R \leq 1\ m \tag{21}$$

where r is the value of the circle's radius that just contains the dynamic obstacle; N is a positive number greater than 1, and the specific value is obtained from experimental. Since the side length of the grid is 1 m, the recognition radius satisfies $R \leq 1\ m$ to avoid the situation that the robot cannot search the path effectively to avoid dynamic obstacles in the case of dense global obstacles.



(a) Robot avoiding traditional obstacles **(b)** Robot avoiding the obstacles in this study

Figure 10. Schematic diagram of dynamic obstacle avoidance.

2) Research on dynamic obstacle avoidance strategy

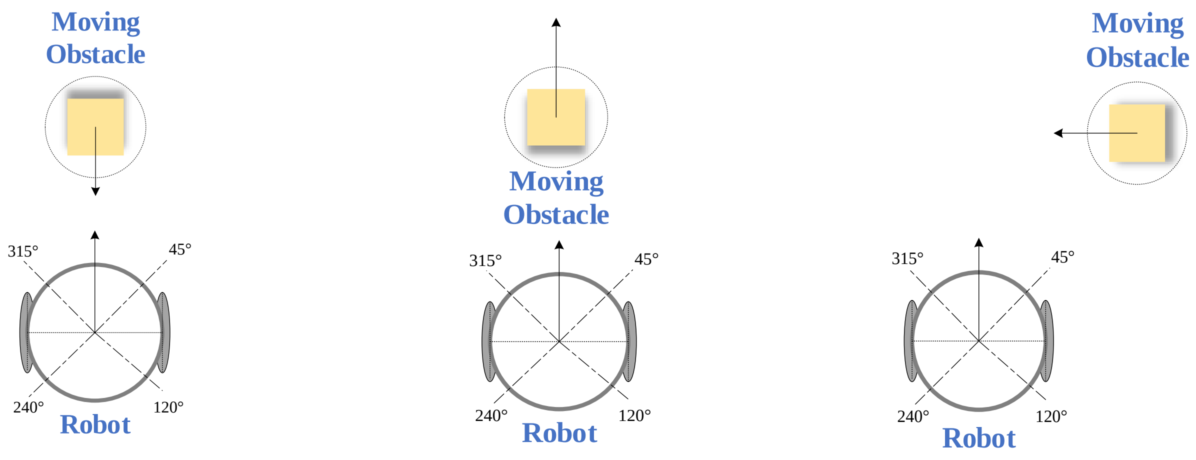
Referring to the research of Liang et al. [43] on the obstacle avoidance scenarios for the unmanned boat with sea surface, we similarly considered multiple types of motion conflicts between the terrestrial robot and dynamic obstacles, and developed the conflict types as well as obstacle avoidance rules, shown in Figure 11 and Table 1 accordingly. After expanding the robot's recognition area of dynamic obstacles, the frontal and rear-end collision problems can be better solved. However, the lateral collision problem (including left collision and right collision) still presents the dilemma shown in Figure 9, for which the following motion constraints are imposed on the robot:

Step 1: When the quantization criteria of robot and obstacle motion direction satisfy the lateral collision scenario, the robot and dynamic obstacle are judged to be in potential motion conflict, based on whether the shortest distance l_r from the end of the robot's predicted trajectory group $tria$ to the obstacle identification region is less than the desired obstacle avoidance distance l_e. If the condition is satisfied, proceed to step 2. If not, the robot performs obstacle avoidance according to our improved DWA.

Step 2: The robot simulates the trajectory group in t_f time period, discarding those speed groups (v, w) and trajectories $tria$ that touch the static obstacle and dynamic obstacle recognition areas.

Step 3: Safe driving distance judgments. Evaluate the optimal trajectory $best(tria)$ according to the evaluation function, and calculate the distance l from the end position of the optimal trajectory $best(tria)$ to the dynamic obstacle recognition area. If $l < l_e$ still exists at this time, the conflict cannot be lifted, and the robot cannot avoid obstacles successfully. Let the optimal trajectory group $tria$ correspond to the velocity group $(v, w) = 0$. Then the robot will stop the motion quickly under the braking constraint.

Step 4: If the distance l from the end position of the optimal trajectory $best(tria)$ to the dynamic obstacle recognition region is greater than the desired obstacle avoidance distance l_e, the conflict is lifted, and the robot resumes motion according to our DWA.



(a) Frontal collision **(b)** Rear-end collision **(c)** Lateral collision

Figure 11. Conflict risk situations.

Table 1. Collision quantification criteria and avoidance rules.

Situation	Quantitative standard	Avoidance rule
frontal collision	$\theta \in [315^\circ, 45^\circ]$	Avoid obstacles
rear-end collision	$\varphi \in [120^\circ, 240^\circ]$	Surpass or follow
right collision	$\theta \in [45^\circ, 120^\circ]$	Avoid obstacles or decelerate to stop
left collision	$\theta \in [240^\circ, 315^\circ]$	Avoid obstacles or decelerate to stop

Note: where θ represents the movement direction of the obstacle towards the robot; φ represents the movement direction of the robot relative to the obstacle.

3.4.3. Multi-robot prioritized obstacle avoidance strategy

The path planning problem of multiple mobile robots is extended from the path planning problem of a single mobile robot. We plan a globally optimal path for each mobile robot in the environment by BAJPSA* algorithm. The DWA obstacle avoidance strategy mainly applies to the static environment or local small-scale dynamic environment (low-speed motion with disturbing obstacles, etc.) [44]. When applied to the multi-robot systems, this does not solve the phenomenon of motion conflict between multiple robots more efficiently [45]. Therefore, we studied the robot's obstacle avoidance strategy for dynamic obstacles through extensive experiments and proposed the dynamic obstacle avoidance rules of 3.4.2. In addition, the priority method [46], as one of the current mainstream techniques for coordinated collision avoidance, can dissipate local conflicts between robots to achieve collision avoidance coordination. To yield better results in global multi-robot motion planning, we combine the BAJPSA* algorithm, improved DWA, and dynamic obstacle avoidance strategy with the multi-mobile robot priority strategy. The algorithm's complexity is simplified by reducing the path planning problem to a dynamic path planning problem with a sequential order for a single mobile robot.

When there are multiple moving robots, the limited environment space is not sufficient for all of them to avoid obstacles. There is an optimal local situation if the robots are treated as dynamic obstacles with the dynamic obstacle avoidance strategy we presented in the previous study. Therefore, we introduce a prioritized strategy and a deceleration mechanism. Different robots have different priorities. When the robot with lower priority encounters one with higher priority and creates a motion conflict, this robot decelerates and stops in advance. The robot with higher priority treats this robot as static obstacle avoidance. As shown in Figure 12: It is assumed that the priority of AGV1, AGV2 and AGV3 decreases in that order. When there is a collision conflict among all three robots, the lower priority AGV2 and AGV3 decelerate to zero. After AGV1 leaves the conflict area, there is still a collision conflict between AGV3 and AGV2. AGV3 stops and waits, resuming motion when AGV2 departs and the collision conflict is lifted.

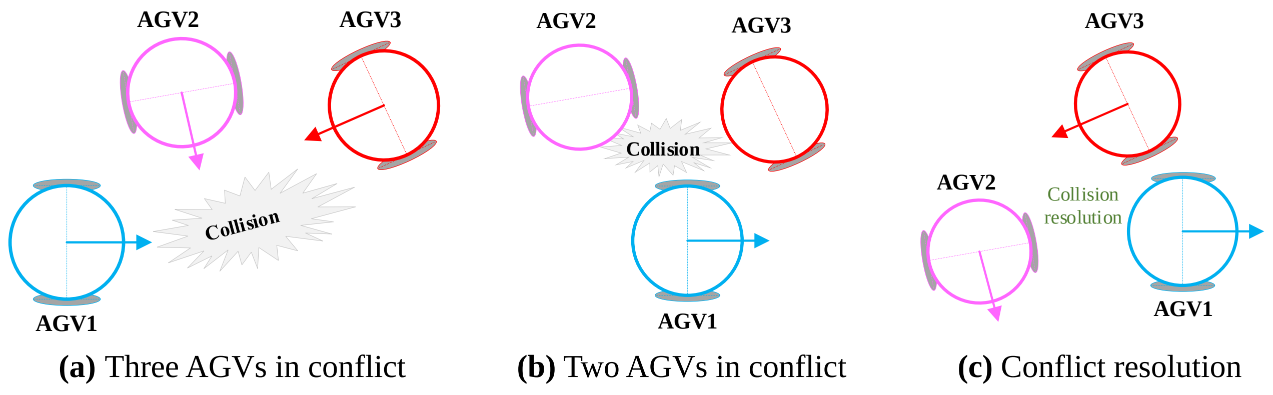
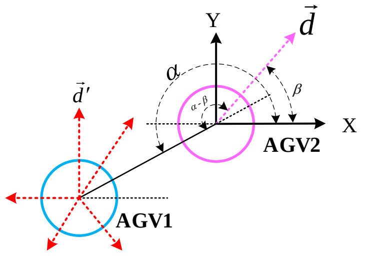
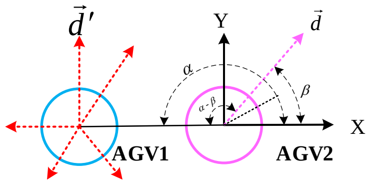
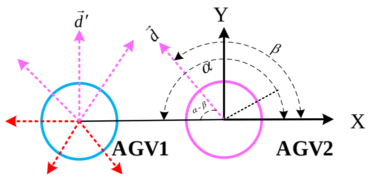
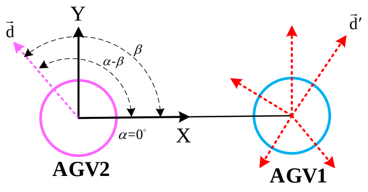
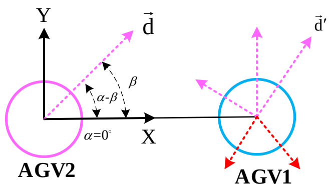
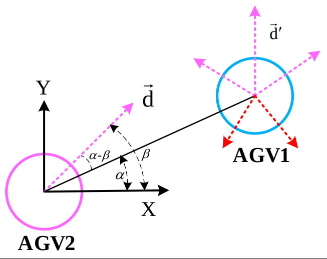


Figure 12. Prioritization avoidance strategy.

The constraints introduced by the robot's kinematic model and environmental obstacles are considered in our study, making the challenge of the multi-machine priority obstacle avoidance strategy focus on specifying the conflict and coordinating the move. Taking two robots with higher priority, AGV1 and lower priority, AGV2, as an example, we investigated multiple conflict types in Table 2 and obtained the following collision conflict judgments and solutions:

Table 2. Multi-robot motion conflict detection and resolution scenarios.

case	Movement Status	Angle Relationship	Potential conflict or not	Conflict resolution strategies
1		$\lvert\alpha-\beta\rvert \geq 90°$	NO	——
2		$\lvert\alpha-\beta\rvert \geq 90°$	NO	——
3		$\lvert\alpha-\beta\rvert < 90°$	YES	AGV1 passes first, AGV2 stops the movement
4		$\lvert\alpha-\beta\rvert \geq 90°$	NO	——
5		$\lvert\alpha-\beta\rvert < 90°$	YES	AGV1 passes first, AGV2 stops the movement
6		$\lvert\alpha-\beta\rvert < 90°$	YES	AGV1 passes first, AGV2 stops the movement

Step 1: If the actual distance d_{12} of two robots is less than the desired obstacle avoidance distance d_e, the potential collision risk.

Step 2: Establish a local coordinate axis with the lower priority robot AGV2 as the center, calculate the angle α between the line of the two robots and the positive direction of the x-axis, and then calculate the relationship between the magnitude of α and the directional angle β of AGV1.

Step 3: Determine whether $|\alpha-\beta|<90°$; if this condition is satisfied, the risk of collision is extremely high. To ensure the safety of multi-robot movement, AGV2 with lower priority decelerates within a short time, according to Eq (22). The AGV1 with higher priority treats AGV2 as an unknown static obstacle and performs obstacle avoidance motion through DWA. When the distance relationship between the two robots is $d_{12} > d_e$ and $|\alpha - \beta| \geq 90°$, AGV2 resumes motion.

In conjunction with the research presented in this study, most experiments show that the proposed prioritized obstacle avoidance strategy applies to most situations. This is mainly attributed to the fact that robots are defined as conflicting motions only when the above conditions are satisfied in terms of distance and angle relationships, and robots that are not in the conflict range perform the corresponding obstacle avoidance strategies based on our DWA.

$$\begin{cases} v(t+1) = v(t) - 2a_v^{max}\Delta t \\ \omega(t+1) = \omega(t) - 2a_\omega^{max}\Delta t \end{cases} \tag{22}$$

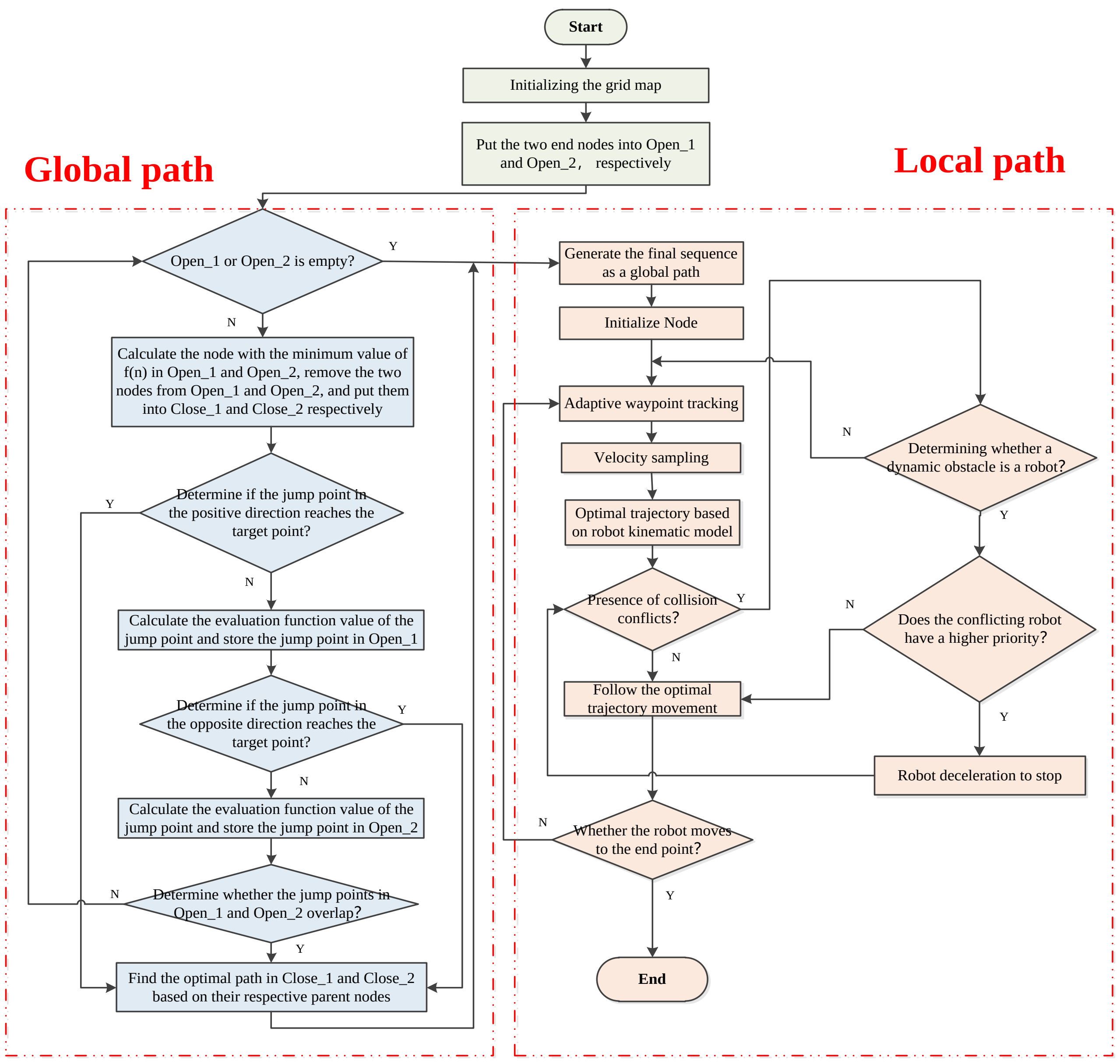

Start
Initializing the grid map
Put the two end nodes into Open_1 and Open_2, respectively
Global path
Local path
Open_1 or Open_2 is empty?
Y
N
Calculate the node with the minimum value of f(n) in Open_1 and Open_2, remove the two nodes from Open_1 and Open_2, and put them into Close_1 and Close_2 respectively
Determine if the jump point in the positive direction reaches the target point?
Calculate the evaluation function value of the jump point and store the jump point in Open_1
Determine if the jump point in the opposite direction reaches the target point?
Calculate the evaluation function value of the jump point and store the jump point in Open_2
Determine whether the jump points in Open_1 and Open_2 overlap?
Find the optimal path in Close_1 and Close_2 based on their respective parent nodes
Generate the final sequence as a global path
Initialize Node
Adaptive waypoint tracking
Velocity sampling
Optimal trajectory based on robot kinematic model
Presence of collision conflicts?
Follow the optimal trajectory movement
Whether the robot moves to the end point?
End
Determining whether a dynamic obstacle is a robot?
Does the conflicting robot have a higher priority?
Robot deceleration to stop

Figure 13. Flow chart of multi-robot path planning.

3.5. Fusion algorithm obstacle avoidance process

We propose a multi-mobile robots obstacle avoidance strategy that guarantees global optimality and safety, which fuses the BAJPSA* algorithm for planning global paths with the DWA local obstacle avoidance algorithm that combines a prioritized obstacle avoidance strategy. The flow of the fusion algorithm is shown in Figure 13.

4) Simulation experiments

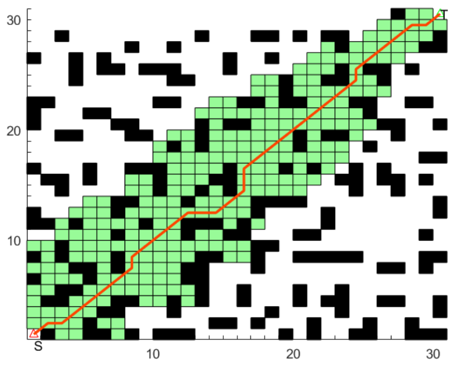
Simulation experiment environment: Windows 10 (64-bit), AMD Ryzen5-4600H with 3GHz, 16 GB memory, the simulation platform is MATLAB R2019a.

The relevant parameters of this paper are as follows: the sampling period $\Delta\boldsymbol{t}$ is 0.1 s; the expected node distance $\boldsymbol{n_{ds}}$is 0.018 $\boldsymbol{m}$; the robot's expected tracking distance $\boldsymbol{d_1}$ is 1.8 $\boldsymbol{m}$; the predicted trajectory to the target point expected distance $\boldsymbol{d_2}$ is 1 $\boldsymbol{m}$; the expected distance parameters $\boldsymbol{l_1}$ and $\boldsymbol{l_3}$ between the robot and the obstacle are 0.4 $\boldsymbol{m}$ and 0.7 $\boldsymbol{m}$ respectively; the maximum error $\boldsymbol{l_2}$ of the robot from the global path is 1 $\boldsymbol{m}$;the expected obstacle avoidance distance $\boldsymbol{l_e}$ of the robot to dynamic obstacles is 1.5 $\boldsymbol{m}$; the expected obstacle avoidance distance $\boldsymbol{d_e}$ between the multiple robots is 2 $\boldsymbol{m}$.The robot model parameters are: maximum linear velocity $\boldsymbol{1\,m/s}$; maximum angular velocity $\boldsymbol{20\,rad/s}$; linear acceleration $\boldsymbol{0.2\,m/s^2}$; angular acceleration $\boldsymbol{5\,rad/s^2}$; linear velocity resolution $\boldsymbol{0.02m/s}$; angular velocity resolution $\boldsymbol{1\,rad/s^2}$ and the evaluation function coefficients are: $\boldsymbol{A = 0.05, B = 0.2, C = 0.3, D = 0}$; the forecast time period $\boldsymbol{t_f}$ is 3.0 $\boldsymbol{s}$.

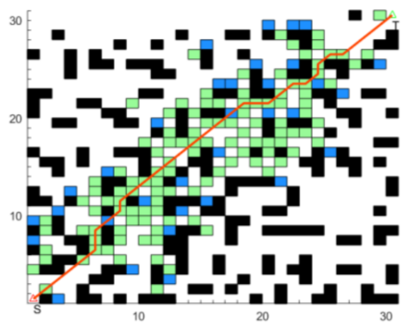
4.1. Global path planning experiments with improved BAJPSA×*

To verify the effectiveness of our BAJPSA*, two sets of maps with scales of 30×30 and 100×100 were selected for simulation and compared with the A* and JPS algorithms. The experimental results and simulation data are shown in Figures 14 and15 and Table 3. The green grid depicts the nodes searched by the algorithm, the blue grid is the forced neighbor nodes of JPS and BAJPSA*, while the red path and the orange path in Figures 14 and 15(c) are the paths of our BAJPSA* forward and the reverse searches, respectively.

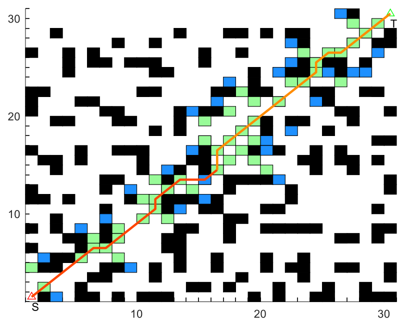
The green node area in Figures 14 and 15 and the number of OPEN and CLOSE list nodes in Table 3 show that our BAJPSA* dramatically reduces the number of extended nodes compared to the A* algorithm, and the improvement is particularly noticeable in a large-scale map of 100×100. There is a 92.5% reduction in the number of extended nodes and a 91.3% reduction in the running time of our BAJPSA* algorithm compared to A*, with a slight increase of 0.585 m in length. There is an advantage of our BAJPSA* over the JPS algorithm, mainly in terms of the search time, which benefits from the mechanism of bidirectional alternating search and the improvement of the heuristic function. We improve the search time of BAJPSA* by 50 % and reduce the number of expansion nodes by 60 % in the 30×30 scale map. Similarly, our BAJPSA* search time for paths is reduced by 79 %, and the number of expansion nodes is slightly improved by 30 % in the 100×100 large-scale map.



(a) A*

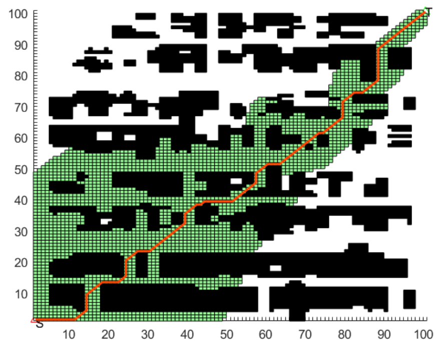


(b) JPSA*

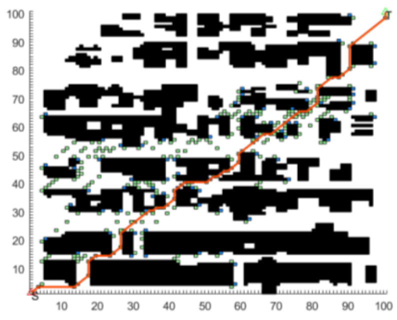


(c) BAJPSA*

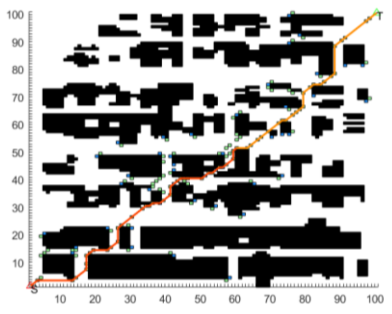
Figure 14. 30×30 environment.



(a) A*



(b) JPSA*



(c) BAJPSA*

Figure 15. 100×100 environment.

Table 3. Comparison of global path planning results.

	30×30 environment			100×100 environment		
Path parameters	A*	JPS	BAJPSA*	A*	JPS	BAJPSA*
Length/m	43.355	43.355	43.355	158.167	158.167	158.752
Run-times/s	0.651	0.863	0.485	6.894	2.855	0.598
Number of path nodes	34	24	27	131	55	89
Number of OPEN list	253	130	51	2509	271	187
Number of CLOSE list	394	98	32	6240	236	121

4.2. Robot obstacle avoidance performance test experiments

4.2.1. Simulation effect test of trajectory tracking capability

To determine the effect of our improved algorithm, the path with the global path length of 17.3616 m marked in black is obtained based on our BAJPSA* algorithm, as shown in Figure 16. The robot with the two-wheel differential motion model is built for trajectory tracking based on our improved DWA, and the test results in Table 4 and the robot angle variation and path error plots in Figure 17 show the experimental data of three robots with different parameters in detail. The specific exploratory analysis is as follows:

In Path_1, we do not consider the robot's new $Path(v,\omega)$ evaluation function, such that the weight $D = 0$. The final path length of the robot travels is 17.7138 m, the robot deviates from the global path by 0.2255 m on average for each movement, and the robot spends 50.8668 s moving from the starting point (1.5, 2.5) to the endpoint (16.5, 10.5).

In Path_2, we set D=0.2, and the final path length of the robot is 17.3676 m, while the average deviation of the robot from the global path is 0.0577 m. Compared with the conventional DWA of Path_1, we sacrifice some efficiency of the algorithm as we simultaneously consider the four Equations indicators (17–20). Moreover, it is evident from the experimental results that the actual robot motion of Path_2 is basically along the global path, except near the obstacles.

In Path_3, we further increase the optimization effect of the $Path(v,\omega)$ indicator, and Path_3 is obtained by setting D = 0.4, which is improved slightly in terms of the path length and path offset indicators compared with Path_2.

Table 4. Robot motion planning results.

Path	Path length/m	Tracking error/m	Search time/s
Path_1	17.7138	0.2255	50.8668
Path_2	17.3676	0.0577	55.8523
Path_3	17.3536	0.0551	61.3893

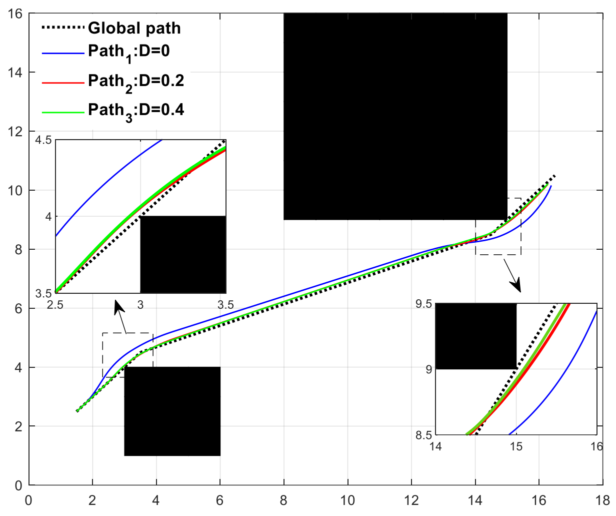
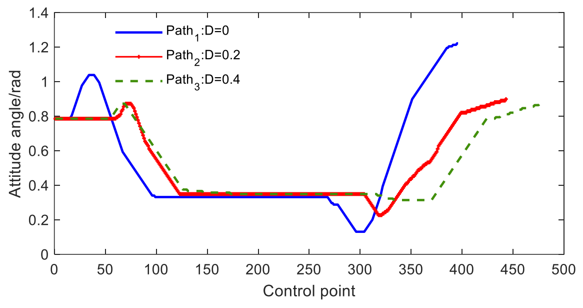
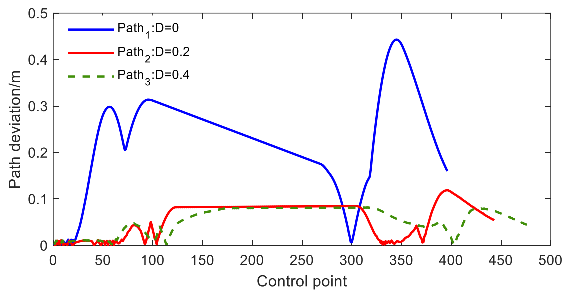


Figure 16. Robot motion planning.



(**a**) Angle change



(**b**) Path tracking error

Figure 17. Comparison of path metrics.

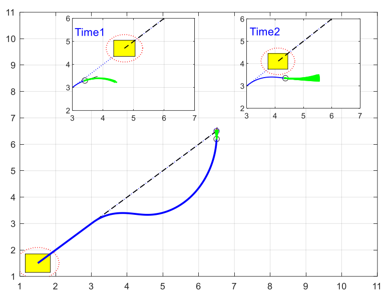
4.2.2. Simulation effect test of dynamic obstacle avoidance capability

To test the effectiveness of our proposed dynamic obstacle avoidance strategy, the following three sets of experiments were conducted for three collision types: frontal, rear-end, and lateral, as shown in Figures 18–21, respectively. The robot model parameters are the same as those in Section 4.2.1.

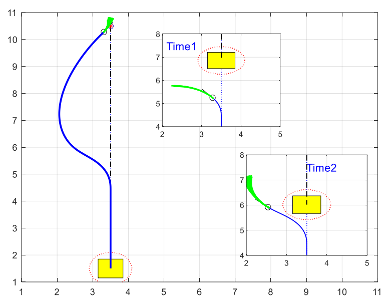
First, we conducted a frontal collision experiment, where the movement speed of dynamic obstacles was set to 0.2 and 0.4 m/s in Figures 18(a),(b), respectively (in Section 3.4.2, we analyzed the actual speed of the robot as $v_r = v_s \cap v_d \cap v_a$, considering the limited search space of the robot, the speed of obstacles in this experiment was less than 0.5 times the maximum speed of the robot). The results show that the robot has a large recognition area for moving obstacles, can avoid them with conflicting frontal motion, and have a specific safety distance. Figure 18(b) shows the environment with high-speed moving obstacles, and the path length of the robot after completing obstacle avoidance is 9.8702 m; it takes 15.62 s to move from the starting point (3.5, 1.5) to the endpoint (3.5, 10.5). The faster the speed of the obstacle movement, the higher the requirement for the robot's obstacle avoidance performance. Thus, the path length of the robot's passage compared to the low-speed moving obstacle in Figure 18(a) increases by 1.9845 m, and the movement duration increases by 1.38 s.

Next, we performed a rear-end collision experiment with an obstacle moving at 0.1 m/s. The test results showed that our robot detected a slow-moving impediment in front and successfully avoided it. The final travel length of the robot was 13.4653 m, and it took 19.82 s.

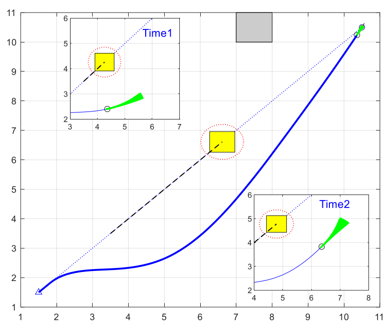
Finally, we set up obstacles with different motion directions and speeds for side collision experiments for three tests, respectively, and added traditional DWA for comparison to reflect the advantages of our obstacle avoidance strategy. In the first group, the speed of obstacle movement is 0.25 m/s, and in the second and third groups is 0.5 m/s. The experimental effect is shown in Figures 19–21. In the first collision experiment shown in Figure 19, the moving speed of the obstacles we set is relatively slow, such that both the traditional DWA and our DWA can successfully avoid obstacles. The travel path lengths of the conventional and our robot are 7.4440 and 7.6449 m, respectively. Keeping the direction of movement of the obstacle, we increased the movement speed of the obstacle to 0.5 m/s and carried out the second collision experiment shown in Figure 20. The travel path lengths of the conventional robot and our robot are 8.1640 m and 7.1840 m, respectively. The experimental results in Figure 21, show that the robot collides with the obstacle under the traditional DWA obstacle avoidance strategy. Our robot can determine conflicting obstacles and slow down and avoid such obstacles that otherwise be avoided successfully. Compared with the first experiment, although our robot has an additional waiting time of 1.68 s, the length of the traveled path is reduced by 0.4609 m, and the safety is guaranteed. For the third test, we changed the direction of movement of this obstacle. From the experimental results of Figure 21, the traditional DWA has the optimal local problem of following the obstacle movement described in Figure 21(a), and the conflict is resolved only when the obstacle stops. The final path length of the robot traveled was 14.8600 m and took 19.16 s. Compared with our robot, the driving distance increases by 1.09 times, and the obstacle avoidance time also increases by 4.17 s.



(**a**) Frontal collision scenario 1

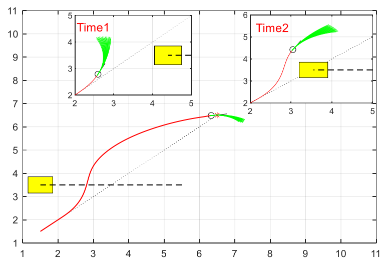


(**b**) Frontal collision scenario 2

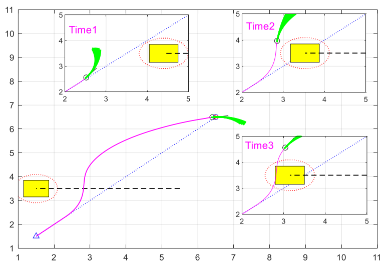


(**c**) Rear-end collision scenario

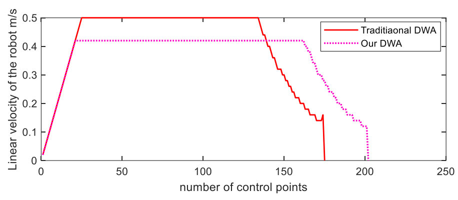
Figure 18. Frontal and rear-end obstacle avoidance tests.



(**a**) Traditional DWA

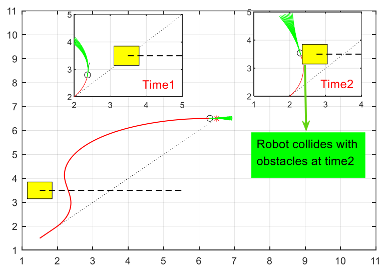


(**b**) Our improved DWA

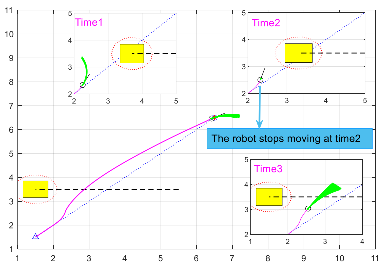


(**c**) Line speed comparison

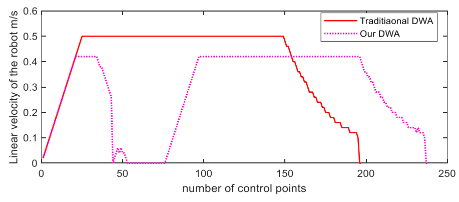
Figure 19. Lateral conflict avoidance test 1.



(**a**) Traditional DWA

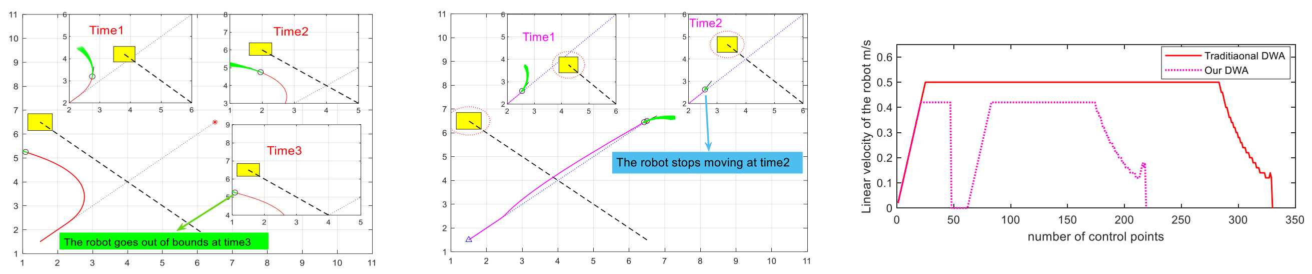


(**b**) Our improved DWA



(**c**) Line speed comparison

Figure 20. Lateral conflict avoidance test 2.



(**a**) Traditional DWA (**b**) Our improved DWA (**c**) Line speed comparison

Figure 21. Lateral conflict avoidance test 3.

4.3. Multi-robot motion planning experiments

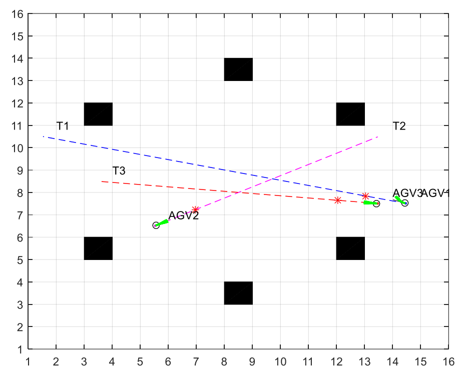
The four experiments with three robots, all shown in Figures 22–28, were performed in four different environments: a globally known static environment, one with unknown static obstacles, one with unknown static and dynamic obstacles, and a dynamic environment with a large-scale sea area, respectively. The effectiveness and stability of our proposed fusion algorithm of the BAJPSA* algorithm, DWA dynamic obstacle avoidance strategy, and multi-robot priority obstacle avoidance strategy have been verified.

4.3.1. Globally known environment

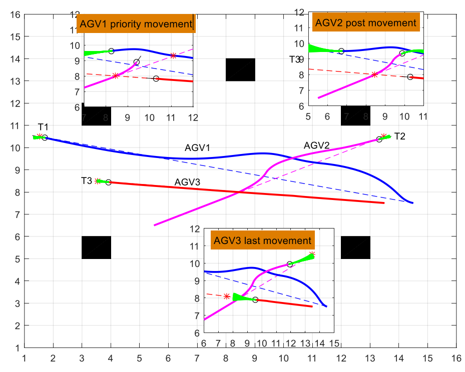
The simulation results for the environment with known global obstacle information are shown in Figures 22–23, where the blue, pink, and red lines are the path situations of the three robots, AGV1, AGV2 and AGV3, respectively; the black grid depicts the static obstacle, for which the robot has a priori information. Figure 22 shows the real-time change of the robot during the planning process. The linear velocity and angular variation curves of the robot are shown in Figures 23(a) and (b), respectively; Figure 23(c) shows the offset of the robot's motion path compared with the global path, where the starting and ending points of AGV1 are (14.5, 7.5) and (1.5, 10.5), respectively; the starting and ending points of AGV2 are (5.5, 6.5) and (13.5, 7.5), respectively; the starting and ending points of AGV3 are (5.5, 6.5) and (13.5, 10.5), respectively.

Figures 22 and 23(a) show that AGV3 decelerates at the 50th motion control node when it detects a collision risk with AGV2, and completely stops at the 80th control node. Then, AGV2 starts decelerating from the 85th control node, until the 95th control node completely stops, as there is a collision conflict with AGV1. AGV1 has the highest priority, such that AGV2 and AGV3 in the stopped state are considered static obstacles, and AGV1 performs reasonable obstacle avoidance based on DWA. Furthermore, Figure 23(c) shows that the actual path of AGV1 movement deviates most from the global path. The collision risk between AGV2 and AGV1 is released, and AGV2 starts to move at the 130th control node. At this time, AGV3 still maintains collision conflict with AGV2. Thus, AGV3 resumes motion at the 160th control node.

Figure 23 shows that AGV3 has the lowest priority, resulting in the most prolonged maintenance state of its stopped motion and less significant changes in the motion direction and path deviation. The additional obstacle avoidance performed by AGV1 and AGV2 for the lower priority robots produced a more noticeable motion angle and global path deviation. However, the whole motion was relatively smooth. The multi-robot motion experiments we conducted took 130.91 s. The travel distances of AGV1, AGV2 and AGV3 were 13.54 m, 8.89 m and 9.63 m, respectively, and the average error of motion deviation from the global path for each motion moment (0.1 s) was 0.3919 m, 0.2056 m, and 0.0120 m, respectively.

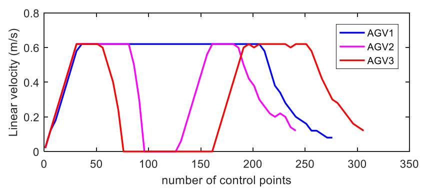


(**a**) Global path planning

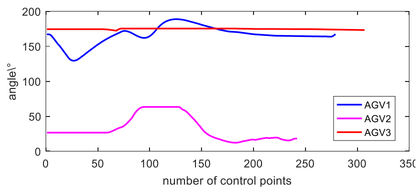


(**b**) Robot local motion

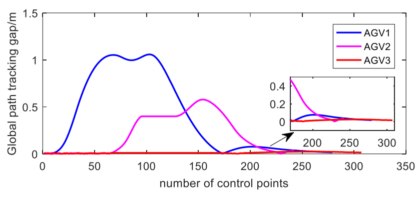
Figure 22. Multi-robot path planning.



(**a**) Line speed



(**b**) Angle of motion



(**c**) Path tracking error

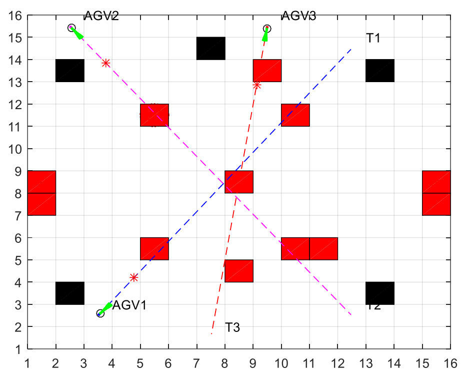
Figure 23. Comparison of path metrics.

4.3.2. Environments containing unknown static obstacles

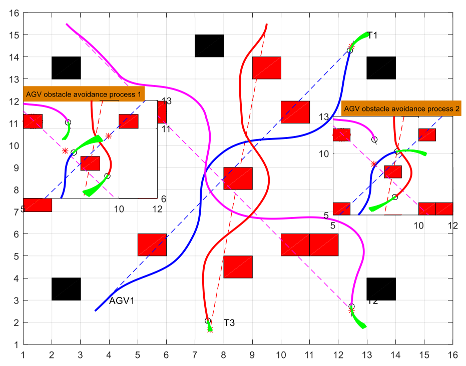
The multi-robot global path based on the BAJPSA* algorithm in the global static environment is shown in Figure 24(a), where the starting and ending points of AGV1 are (3.5, 2.5) and (12.5, 14.5), respectively; the starting and ending points of AGV2 are (2.5, 15.5) and (12.5, 2.5), respectively; the starting and ending points of AGV3 are (9.5, 15.5) and (7.5, 1.5), respectively. Then, the randomly distributed unknown static obstacles (the red grid) were increased to conduct the following simulation experiments.

The results shown in Figures 24 and 25 indicate that our multi-robot systems can avoid random static obstacles successfully when moving along the global path in an environment that includes unknown factors. The risk of collision between multi-robot systems is solved successfully with an improved prioritization strategy. What we know from Figures 24(b) and 25(a) is that when the higher priority AGV2 collides with lower priority AGV1, AGV2 decelerates from the 140th control node, and the velocity reaches zero at the 160th control node. The conflict is released once AGV1 moves away from AGV2 and AGV2 resumes motion at the 180th control node and successfully avoids random obstacles.

The variation of the motion speed, travel direction, and global path offset for each robot is relatively significant, as the obstacles of unknown disturbance were considered compared to those in Section 4.3.1. In conclusion, the travel distances for AGV1, AGV2 and AGV3 are 16.36, 19.46 and 14.57 m, respectively. The average values of global path offset for each robot movement are 0.5353, 0.7853, and 0.4287 m, respectively. The robot movement took 160.68 s program running time.

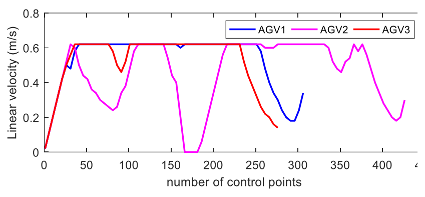


(**a**) Global path planning

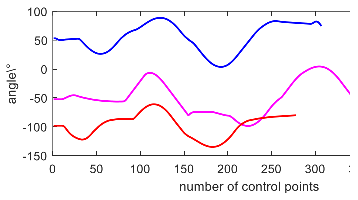


(**b**) Robot local motion

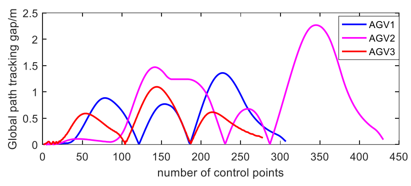
Figure 24. Multi-robot path planning.



(**a**) Line speed



(**b**) Angle of motion



(**c**) Path tracking error

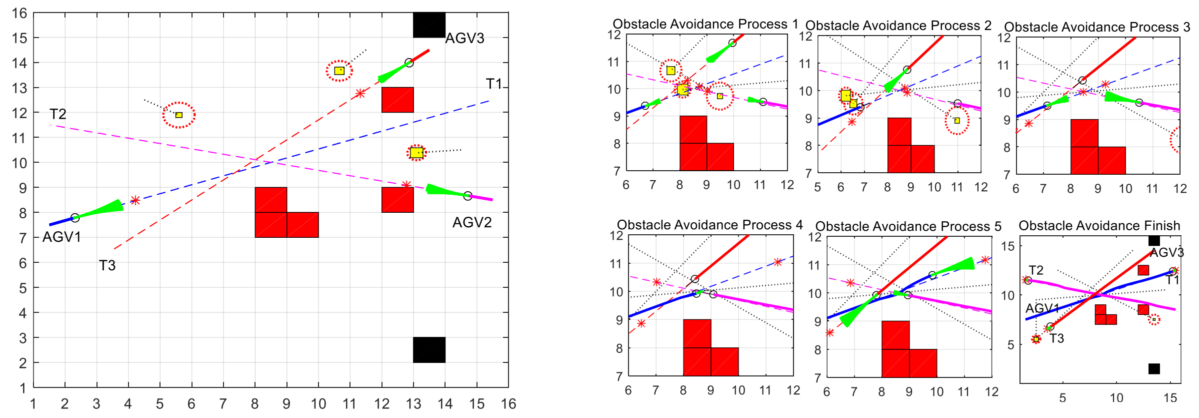
Figure 25. Comparison of path metrics.

4.3.3. Environments containing unknown dynamic obstacles

There are several dynamic obstacles we added to the unknown static obstacles in Section 4.3.2 to further test the applicability of single-robot dynamic obstacle avoidance strategy and multi-robot prioritized obstacle avoidance strategy in the given scenario, where the starting and ending points of AGV1 are (1.5, 7.5) and (15.5, 12.5), respectively; the starting and ending points of AGV2 are (15.5, 8.5) and (1.5, 12.5), respectively; the starting and ending points of AGV3 are (13.5, 14.5) and (3.5, 6.5), respectively. The yellow squares shown in Figure 26 are dynamic obstacles without a priori knowledge for the robots, and the red enclosures are the recognizable regions that the robots are assigned. The study of dynamic obstacle motion speed and robot recognition area in Section 3.4.2 indicates that they are positively proportional. Therefore, we set the movement speed for the three dynamic obstacles of the small, medium, and large sizes as $0.5, 0.39, and\ 0.30\ m/s$, respectively; the radius of the recognition area given is 0.55, 0.40 and 0.35 m, respectively; the preset motion path is shown in the last figure of Figure 26.

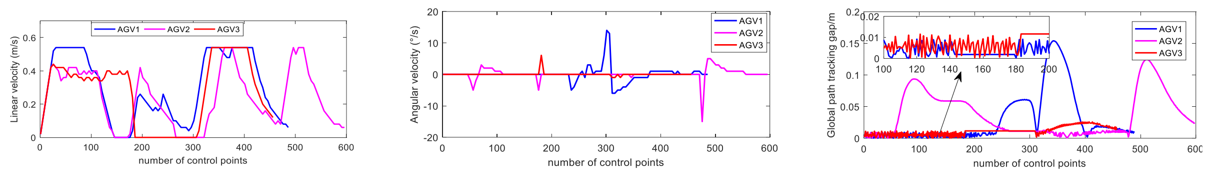
As shown in Figure 26(b), three robots and three unknown dynamic obstacles are encountered at the center of the map. Figures 26(b) and 27(a) show that AGV1 and AGV2 detect dynamic obstacles in front of them, and cannot avoid them because of the limited environment and rapid movement of the obstacles. According to the dynamic obstacle avoidance rules for motion conflicts in Section 3.4.2, the deceleration of both AGV1 and AGV2 starts from around the 125th control node and stops entirely at the 150th control node. After the dynamic obstacle leaves, both AGV1 and AGV2 resume motion around the 185th control node. Subsequently, the collision risk with AGV1 is detected by AGV3 and AGV2 at the 170th and 200th control nodes, respectively. According to Section 3.4.3, AGV3 and AGV2 decelerate and wait until the conflict is removed in the multi-robot priority obstacle avoidance strategy.

This experimental simulation result demonstrates the effectiveness of our improved single-robot dynamic obstacle avoidance strategy combined with a multi-robot priority avoidance strategy in an environment with random static and unknown dynamic obstacles. As we find from Figures 27(b) and (c), there is obstacle avoidance in AGV1, resulting in a large angle and path offset. The motion distances of the robot are 14.66, 14.11 and 12.38 m, respectively; the errors of the robot's single-step motion offsetting the global path are 0.0288, 0.0361 and 0.0114 m, respectively; the total running time of the algorithm is 292.0472 s.



(**a**) Global path planning (**b**) Robot local motion

Figure 26. Multi-robot path planning.



(**a**) Line speed (**b**) Angle of motion (**c**) Path tracking error

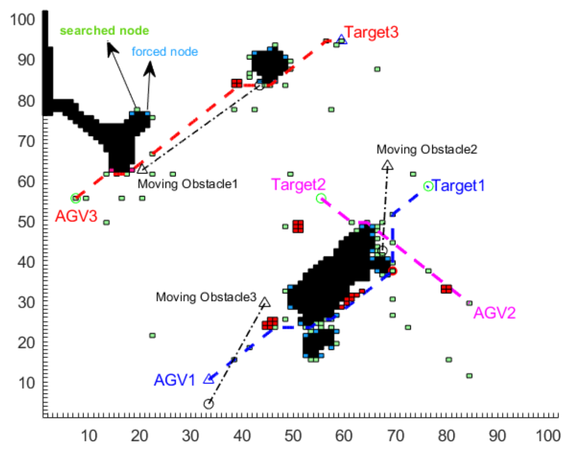
Figure 27. Comparison of path metrics.

4.3.4. Dynamic environment of large-scale sea area

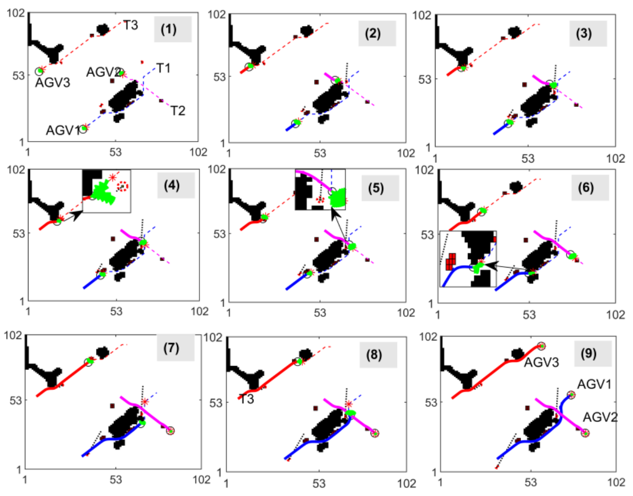
The multi-robot global path based on the BAJPSA* algorithm in a large-scale sea environment with an accuracy of 10 m is shown in Figure 28(a), where the starting and ending points of AGV1 are (7.5, 55.5) and (59.5, 94.5), respectively; the starting and ending points of AGV2 are (55.5, 55.5) and (84.5, 29.5), respectively; the starting and ending points of AGV3 and endpoints are (33.5, 10.5) and (76.5, 58.5), respectively. The BAJPSA* algorithm takes 0.3328, 0.0972 and 0.0975 s to plan the global path for the three robots; the numbers of extended nodes are 41, 22 and 31, respectively. The global path lengths are 531.481, 323.201 and 636.734 m, respectively. Then, unknown static and dynamic obstacles with random distribution are added, and the following simulation experiments are conducted.

Figure 28 shows that our fusion algorithm is equally effective in the large-scale map environment. As indicated in Figure 28(b-4), AGV3 encounters a dynamic obstacle traveling in the same direction. It overtakes left side to avoid the obstacle traveling ahead, following the dynamic obstacle avoidance rules of the single robot. In Figure 28(b-6), AGV1 successfully avoids the random static obstacle and traverses the map's narrow area. In Figure 28(b-7), AGV2 and AGV3 successfully avoid the random static obstacles. Figure 28(b-9) shows the final travel paths of the three robots. The traces of the robots are smooth and fit the global path.

The experimental simulation results demonstrate the effectiveness of our multi-robot obstacle avoidance strategy with BAJPSA* fusion improved DWA in a large-scale environment. The motion distances of AGV1, AGV2 and AGV3 are 693 m, 388.5 m and 667 m, respectively, and the average errors of motion deviation from the global path at each motion moment (0.1 s) are 4.052 m, 0.895 m and 0.755 m, respectively.



(a) Global path planning



(b) Robot local motion

Figure 28. Multi-robot path planning.

5. Conclusions

To solve the path planning problem of distributed multiple robots in dynamic environments, we propose a BAJPSA* algorithm fused with adaptive DWA, performing in two stages.

In the first stage, we plan the globally optimal path for each robot by the BAJPSA* algorithm, with simulation results demonstrating the effectiveness of BAJPSA* in global path planning. In the second stage, we perform the local path planning. The adaptive navigation strategy and path deviation evaluation function are proposed to improve the path tracking capability of the traditional DWA. Next, we categorize and discuss multiple unknown static and dynamic obstacle environments with motion conflict scenarios, and propose dynamic obstacle avoidance rules for the single robot. Then, we extend the single-robot to distributed multi-robot with decision rights, discuss multiple classes of motion conflict situations, and achieve cooperative multi-robot avoidance by fusing the prioritizing avoidance rules. The simulation results demonstrate the effectiveness of this algorithm for multi-robot path planning in unknown dynamic environments.

In this study, unknown static, as well as highly dynamic environments are the environments we focus on, and more complex factors (non-flat terrain, large-scale robots, etc.) will be gradually considered in future work. We can also test this algorithm in a robot platform and further on the multi-robot cooperative efficiency problem.

Acknowledgments

This research was funded by the National Key R&D Program of China, grant number 61163051.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. F. Rubio, F. Valero, C. Llopis-Albert, A review of mobile robots: Concepts, methods, theoretical framework, and applications, *Int. J. Adv. Rob.Syst.*, **16** (2019), 1729881419839596. https://doi.org/10.1177/1729881419839596
2. S. J. Fusic, G. Kanagaraj, K. Hariharan, S. Karthikeyan, Optimal path planning of autonomous navigation in outdoor environment via heuristic technique, *Transp. Res. Interdiscip. Perspect.*, **12** (2021), 100473. https://doi.org/10.1016/j.trip.2021.100473
3. J. Li, J. Sun, L. Liu, J. Xu, Model predictive control for the tracking of autonomous mobile robot combined with a local path planning, *Meas. Control*, **54** (2021), 1319–1325. https://doi.org/10.1177/00202940211043070
4. A. V. Le, V. Prabakaran, V. Sivanantham, R. E. Mohan, Modified a-star algorithm for efficient coverage path planning in tetris inspired self-reconfigurable robot with integrated laser sensor, *Sensors*, **18** (2018), 2585. https://doi.org/10.3390/s18082585
5. H. Wang, X. Qi, S. Lou, J. Jing, H. He, W. Liu, An efficient and robust improved A* algorithm for path planning, *Symmetry*, **13** (2021), 2213. https://doi.org/10.3390/sym13112213
6. B. Zhang, D. Zhu, A new method on motion planning for mobile robots using jump point search and Bezier curves, *Int. J. Adv. Robot. Syst.*, **18** (2021), 17298814211019220. https://doi.org/10.1177/17298814211019220
7. F. H. Ajeil, I. Ibraheem, A. T. Azar, A. J. Humaidi, Grid-based mobile robot path planning using aging-based ant colony optimization algorithm in static and dynamic environments, *Sensors*, **20** (2020), 1880. https://doi.org/10.3390/s20071880

8. C. Miao, G. Chen, C. Yan, Y. Wu, Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm, *Comput. Indust. Eng.*, **156** (2021), 107230. https://doi.org/10.1016/j.cie.2021.107230
9. B. Song, Z. Wang, L. Zou, An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve, *Appl. Soft Comput.*, **100** (2021), 106960. https://doi.org/10.1016/j.asoc.2020.106960
10. X. Guo, M. Ji, Z. Zhao, W. Zhang, Global path planning and multi-objective path control for unmanned surface vehicle based on modified particle swarm optimization (PSO) algorithm, *Ocean Eng.*, **216** (2020), 107693. https://doi.org/10.1016/j.oceaneng.2020.107693
11. M. A. Hossain, I. Ferdous, Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, *Robot. Auton. Syst.*, **64** (2015),137–141. https://doi.org/10.1016/j.robot.2014.07.002
12. Y. P. Chen, Y. Li, G. Wang, Y. F. Zheng, Q. Xu, J. H. Fan, et al., A novel bacterial foraging optimization algorithm for feature selection, *Expert Syst. Appl.*, **83** (2017), 1–17. https://doi.org/10.1016/j.eswa.2017.04.019
13. H. Tang, W. Sun, H. Yu, A. Lin, M. Xue, A multirobot target searching method based on bat algorithm in unknown environments, *Expert Syst. Appl.*, **141** (2020), 112945. https://doi.org/10.1016/j.eswa.2019.112945
14. G. G. Wang, H. E. Chu, S. Mirjalili, Three-dimensional path planning for UCAV using an improved bat algorithm, *Aerosp. Sci. Technol.*, **49** (2016), 231–238. https://doi.org/10.1016/j.ast.2015.11.040
15. Z. Yan, J. Zhang, J. Zeng, J. Tang, Three-dimensional path planning for autonomous underwater vehicles based on a whale optimization algorithm, *Ocean Eng.*, **250** (2022), 111070. https://doi.org/10.1016/j.oceaneng.2022.111070
16. F. Gul, I. Mir, L. Abualigah, S. Mir, M. Altalhi, Cooperative multi-function approach: A new strategy for autonomous ground robotics, *Future Gener. Comput. Syst.*, **134** (2022), 361–373. https://doi.org/10.1016/j.future.2022.04.007
17. D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, E. Gunawan, A systematic literature review of A* pathfinding, *Proc. Comput. Sci.*, **179** (2021), 507–514. https://doi.org/10.1016/j.procs.2021.01.034
18. Q. Wu, Z. Chen, L. Wang, H. Lin, Z. Jiang, S. Li, et al., Real-time dynamic path planning of mobile robots: A novel hybrid heuristic optimization algorithm, *Sensors*, **20** (2020), 188. https://doi.org/10.3390/s20010188
19. L. Chang, L. Shan, Y. Dai, Multi-robot formation control in unknown environment based on improved DWA, *Control Decis.*, (2021), 1–10. https://doi.org/10.13195/j.kzyjc.2020.1817.
20. J. Sun, G. Liu, G. Tian, J. Zhang, Smart obstacle avoidance using a danger index for a dynamic environment, *Appl. Sci.*, **9** (2019),1589. https://doi.org/10.3390/app9081589
21. L. Chang, L. Shan, C. Jiang, Y. Dai, Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment, *Auton. Robot.*, **45** (2021), 51–76. https://doi.org/10.1007/s10514-020-09947-4
22. Z. Lin, M. Yue, G. Chen, J. Sun, Path planning of mobile robot with PSO-based APF and fuzzy-based DWA subject to moving obstacles, *Trans. Inst. Meas. Control*, **44** (2022), 121–132. https://doi.org/10.1177/01423312211024798
23. Y. Chen, G. Luo, Y. Mei, J. Yu, X. Su, UAV path planning using artificial potential field method updated by optimal control theory, *Int. J. Syst. Sci.*, **47** (2016), 1407–1420. https://doi.org/10.1080/00207721.2014.929191

24. U. Orozco-Rosas, O. Montiel, R. Sepúlveda, Mobile robot path planning using membrane evolutionary artificial potential field, *Appl. Soft Comput.*, **77** (2019), 236–251. https://doi.org/10.1016/j.asoc.2019.01.036
25. X. Zhong, J. Tian, H. Hu, X. Peng, Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment, *J. Intell. Robot. Syst.*, **99** (2020), 65–77. https://doi.org/10.1007/s10846-019-01112-z
26. X. Ji, S. Feng, Q. Han, H. Yin, S. Yu, Improvement and fusion of A* algorithm and dynamic window approach considering complex environmental information, *Arab. J. Sci. Eng.*, **46** (2021), 7445–7459. https://doi.org/10.1007/s13369-021-05445-6
27. Z. Wang, G. Li, J. Ren, Dynamic path planning for unmanned surface vehicle in complex offshore areas based on hybrid algorithm, *Comput. Commun.*, **166** (2021), 49–56. https://doi.org/10.1016/j.comcom.2020.11.012
28. B. Sahu, P. K. Das, M. Kabat, Multi-robot cooperation and path planning for stick transporting using improved Q-learning and democratic robotics PSO, *J. Comput. Sci.*, **60** (2022), 101637. https://doi.org/10.1016/j.jocs.2022.101637
29. Y. Dai, Y. Kim, S. Wee, D. Lee, S. Lee, A switching formation strategy for obstacle avoidance of a multi-robot system based on robot priority model, *ISA Trans.*, 5**6** (2015), 123–134. https://doi.org/10.1016/j.isatra.2014.10.008
30. H. Sang, Y. You, X. Sun, Y. Zhou, F. Liu, The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations, *Ocean Eng.*, **23** (2021), 108709. https://doi.org/10.1016/j.oceaneng.2021.108709
31. P. K. Das, H. S. Behera, B. K. Panigrahi, A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning, *Swarm Evol. Comput.*, **28** (2016), 14–28. https://doi.org/10.1016/j.swevo.2015.10.011
32. P. K. Das, P. K. Jena, Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators, *Appl. Soft Comput.*, **92** (2020), 106312. https://doi.org/10.1016/j.asoc.2020.106312
33. R. K. Dewangan, A. Shukla, W. W. Godfrey, A solution for priority-based multi-robot path planning problem with obstacles using ant lion optimization, *Mod. Phys. Lett. B*, **34** (2020), 2050137. https://doi.org/10.1142/S0217984920501377
34. J. M. Yang, C. M. Tseng, P. S. Tseng, Path planning on satellite images for unmanned surface vehicles, *Int. J. Naval Archit. Ocean Eng.*, **7** (2015), 87–99. https://doi.org/10.1515/ijnaoe-2015-0007
35. L. Yang, L. Fu, P. Li, J. Mao, N. Guo, L. Du, LF-ACO: An effective formation path planning for multi-mobile robot, *Math. Biosci. Eng.*, **19** (2022), 225–252. https://doi.org/10.3934/mbe.2022012
36. D. Harabor, A. Grastien, Online graph pruning for pathfinding on grid maps, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **25** (2011), 1114–1119. https://doi.org/10.1609/aaai.v25i1.7994
37. D. Harabor, A. Grastien, Improving jump point search, in *Proceedings of the International Conference on Automated Planning and Scheduling*, **24** (2014), 128–135.
38. C. Li, X. Huang, J. Ding, K. Song, S. Lu, Global path planning based on a bidirectional alternating search A* algorithm for mobile robots, *Comput. Indust. Eng.*, **168** (2022), 108123. https://doi.org/10.1016/j.cie.2022.108123

39. Y. Singh, S. Sharma, R. Sutton, D. Hatton, A. Khan, A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents, *Ocean Eng.*, **169** (2018), 187–201. https://doi.org/10.1016/j.oceaneng.2018.09.016
40. L. Yang, L. Fu, P. Li, J. Mao, N. Guo, An effective dynamic path planning approach for mobile robots based on ant colony fusion dynamic windows, *Machines*, **10** (2022), 50. https://doi.org/10.3390/machines10010050
41. S. M. H. Rostami, A. K. Sangaiah, J. Wang, X. Liu, Obstacle avoidance of mobile robots using modified artificial potential field algorithm, *EURASIP J. Wirel. Commun. Netw.*, **2019** (2019), 1–19. https://doi.org/10.1186/s13638-019-1396-2
42. E. A. Torkamani, Z. Xi, Systematical collision avoidance reliability analysis and characterization of reliable system operation for autonomous navigation using the dynamic window approach, *ASCE-ASME J. Risk Uncertainty Eng. Syst., Part B*, **8** (2022), 031106. https://doi.org/10.1115/1.4053941
43. C. Liang, X. Zhang, Y. Watanabe, Y. Deng, Autonomous collision avoidance of unmanned surface vehicles based on improved A star and minimum course alteration algorithms, *Appl. Ocean Res.*, **113** (2021),102755. https://doi.org/10.1016/j.apor.2021.102755
44. M. Kobayashi, N. Motoi, Local path planning: Dynamic window approach with virtual manipulators considering dynamic obstacles, *IEEE Access*, **10** (2022), 17018–17029. https://doi.org/10.1109/ACCESS.2022.3150036
45. E. Olcay, F. Schuhmann, B. Lohmann, Collective navigation of a multi-robot system in an unknown environment, *Robot. Auton. Syst.*, **132** (2020), 103604. https://doi.org/10.1016/j.robot.2020.103604
46. L. Gracia, A. Sala, F. Garelli, Robot coordination using task-priority and sliding-mode techniques, *Robot. Comput. Integr. Manuf.*, **30** (2024), 74–89. https://doi.org/10.1016/j.rcim.2013.08.003