IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 August 2022, accepted 7 September 2022, date of publication 12 September 2022,
date of current version 26 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3206356

RESEARCH ARTICLE

ASL-DWA: An Improved A-Star Algorithm for Indoor Cleaning Robots

HAOXIN LIU AND YONGHUI ZHANG
School of Information and Communication Engineering, Hainan University, Haikou 570288, China
Corresponding author: Haoxin Liu (369818556@qq.com)

ABSTRACT The traditional A-star algorithm has many search nodes, and the obtained path has polylines and cannot avoid local unknown obstacles. In response to these problems, this paper proposes a new improved A-star algorithm suitable for indoor cleaning robots, called ASL-DWA (A Star Leading Dynamic Window Approach). First, to solve the problem of many search nodes in the A-star algorithm, a new hybrid heuristic function that combines Euclidean distance and point-to-line distance is proposed, thereby reducing the number of search nodes. Then, to solve the problem that the A-star algorithm has a polyline path and cannot avoid local unknown obstacles, this paper designs the global path yaw angle according to the relationship between the real-time position of the robot and the global path, which is added as a score item to the traditional score function. A decay coefficient with prediction function is also added to the score function to reduce the risk of the algorithm falling into local optima. Finally, a mechanism to adaptively adjust the coefficient according to the distance between the robot and the target point is designed, thereby realizing ASL-DWA. The ASL-DWA algorithm is tested in three indoor environments and compared with traditional algorithms. The experimental results show that ASL-DWA can meet the path planning requirements of mobile robots in indoor environments, and has obvious advantages over traditional algorithms.

INDEX TERMS Indoor cleaning robot, A-star algorithm, hybrid heuristic function, global path yaw angle, adaptive weighted score function.

I. INTRODUCTION

In recent years, with the rapid development of mobile robot technology and the increase in labor costs, mobile robots are increasingly applied to various fields. Among them, the indoor cleaning robot has become one of several mobile robot products with the largest output and the highest market penetration rate due to its relatively low production cost and its advantages of being closer to people's daily life [1]. Indoor cleaning robots can autonomously complete functions such as map construction, path planning, garbage collection, and charging in the home environment [2]. With the popularization and application of cleaning robots, higher and higher performance requirements are put forward. Among them, path planning is a core technology in the field of cleaning robots, and it is also a key technology to improving user experience [3].

Simultaneous Localization and Mapping (SLAM) technology fuses multi-sensor information to obtain an environment map [4], and the mobile robot performs path planning on this basis. The core goal of path planning is to find a path from the starting point to the goal point, which is safe and has the least cost [5].

Global path planning is a type of static planning that plans an optimal path on a known global map [6]. The classic global path planning algorithms include the A-star algorithm, D-star algorithm, Rapidly-exploring Randon Tree (RRT) [7], Genetic Algorithm (GA) [8], and Ant Colony algorithm Optimization (ACO) [9], and so on. Among them, the A-star algorithm is considered to be one of the most effective algorithms for solving the shortest path in static maps due to its high planning efficiency [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Liu.

However, when the A* algorithm is used for path planning in a complex indoor environment, more search nodes are required, and polyline paths are generated, which reduces the efficiency of path planning. Some improved algorithms based on A-stars are studied.

The bidirectional A-star algorithm [11] expands from both the starting point and the target point at the same time, which improves the convergence speed of the algorithm. Breadth-first search (BFS) BFS algorithm uses queues to achieve node expansion [12], and depth-first search (DFS) uses a recursive approach [13], which effectively reduces the number of search nodes in environments with fewer obstacles. Fu *et al.* proposed an improved A-star algorithm to shorten the path by judging whether there are obstacles between the current node and the target point [14]. Yan *et al.* connected the two adjacent nodes before and after the current node on the path of A-star and deleted the current node if there was no obstacle in the middle, thereby reducing the number of nodes [15]. The above two methods are computationally intensive and cannot smooth the turning path. Song *et al.* [16] used three smoothers to reduce the number of turning points and removed some redundant path nodes, but this method is susceptible to the number of nodes and requires multiple iterations. Liu *et al.* [17] combined the Delaunay triangulation method with the A-star algorithm to reduce the search range of the A-star algorithm, but this method requires an additional calculation of the Delaunay triangulation, and the path planning efficiency is low. Tang *et al.* [18] set the filter function to avoid the turning angle of the path obtained by the A-star from being too large and combined the cubic B-spline interpolation algorithm to smooth the path. This method has the problem that the filter function threshold is not easy to select. Kai *et al.* improved the success rate by adding heuristic-based stagnation detection to each expansion node and introduced a predefined unacceptable heuristic when detecting that the algorithm was not moving towards the goal [19]. This method improves the speed of the A-star search, but when to switch multiple heuristics is a problem.

The above studies have improved the A* algorithm to a certain extent. However, mobile robots often encounter unknown obstacles in indoor environments, that is, obstacles that are not detected on the static map. The above algorithms cannot guide the robot to avoid these local obstacles.

The local path planning algorithm is a kind of method in a dynamic environment. It dynamically plans the path according to the robot's motion model, real-time position, real-time obstacle distribution, and other factors, so it can guide the robot to avoid unknown obstacles [20], [21], [22]. Common local path planning algorithms are Dynamic-Window Method (DWA) [23], Time Elastic Band (TEB) [24], Model Predictive Control (MPC) [25], and so on. Compared with other algorithms, the DWA algorithm has the advantages of low computational complexity, conformity to robot kinematics, and strong flexibility, so it is widely used. However, in the indoor environment, the distribution of obstacles is relatively dense, and the local path planning algorithm is easy to falls into the local optimum, resulting in the inability to reach the target point.

Aiming at the cleaning robot in the indoor environment, this paper proposes an improved A-star path planning method, called ASL-DWA. In the first stage, the A-star algorithm with a new hybrid heuristic function is used to obtain the global path, during which the safety distance between the robot and the obstacle is taken into account. In the second stage, the global path yaw angle is designed, which describes the relationship between the robot's real-time pose and the global path. In the third stage, the global path yaw angle is added to the traditional scoring function as a scoring term. In the fourth stage, attenuation coefficients with prediction function are used. Finally, an adaptive mechanism that autonomously adjusts the various score coefficients according to the distance between the robot and the target point is used to realize ASL-DWA.

The contributions of this paper are as follows:

1) A hybrid heuristic function based on Euclidean distance and point-to-line distance is proposed, thereby reducing the number of search nodes in the A-star algorithm.
2) The global path yaw angle is proposed, which combines the global path information with the real-time information of the robot so that the robot can avoid unknown obstacles and is not easy to fall into the local optimum. At the same time, the method also avoids the appearance of polyline paths, thereby reducing the movement time of the robot from the starting point to the target point.
3) A decay coefficient with a prediction function is proposed. The score is intervened by predicting the closest distance between the robot and the obstacle in the next few cycles, thereby further avoiding the algorithm from falling into a local optimum.
4) Propose an adaptive weighting mechanism. By judging the distance between the robot and the target point, the weights of each score are adaptively adjusted. This mechanism enables the robot to avoid obstacles effectively and reach the target point accurately.

The rest of this paper is organized as follows. The second section introduces the A-star algorithm based on the new hybrid heuristic function, including environment modeling, the design principle of the hybrid heuristic function, and the selection of the expansion threshold. Section III introduces the design principles of the global path yaw angle, attenuation coefficient, and adaptive mechanism. Then a new scoring function is introduced on this basis, and finally, the algorithm flow of ASL-DWA is introduced. The fourth section is the experiment and results in analysis. First, the proposed hybrid heuristic function is compared with the other four traditional heuristic functions, and then ASL-DWA is compared with several other algorithms. Section 5 summarizes the performance advantages of the ASL-DWA algorithm and the shortcomings of current research, and looks forward to several directions for future research.

II. HYBRID HEURISTIC FUNCTION BASED A-STAR ALGORITHM

A. ENVIRONMENT MODELING

The cleaning robot used in this paper is shown in Fig 1. The robot has a circular shape and a diameter of 34cm.

FIGURE 1. The cleaning robot.

The cleaning robot moves under the drive of two driving wheels and uses SLAM technology to integrate multi-sensor information such as laser scanners, gyroscopes, and odometers to build a grid map of the indoor environment, as shown in Fig 2.

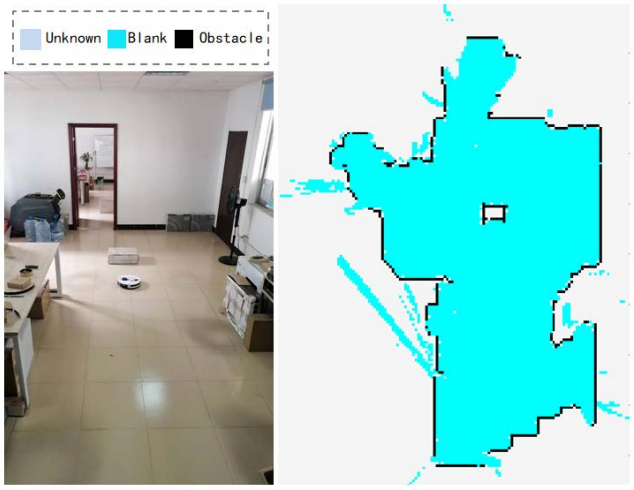

FIGURE 2. Environment modeling.

Fig 2.a is the working environment of the cleaning robot, and Fig 2.b is the grid map obtained by the cleaning robot in this environment. In Fig 2.b, each grid corresponds to an area of 5cm∗5cm. The black grid represents the obstacle area; the cyan grid represents the blank area; the smoke-white grid represents the unknown area, that is, the area that has not been detected by the robot.

The A-star algorithm performs path planning in a grid map. The resulting path should allow the cleaning robot to pass safely, so the robot's outer dimensions should be considered. At the same time, to ensure the safety of the robot, the outside of the robot should maintain a sufficient safety distance from obstacles. For these reasons, this paper identifies the area near the obstacle on the map, which is called the warning area, that is, the area that is not recommended for robots to pass through although there are no obstacles.

As shown in Fig. 3(a), the area with the obstacle net P as the center and r+d as the radius is marked as the warning area, where r is the radius of the robot and d is the minimum safe distance between the outside of the robot and the obstacle. The final result is shown in Fig 3.b. In the figure, cyan indicates the safe passage area; black indicates obstacles; green indicates the warning area.

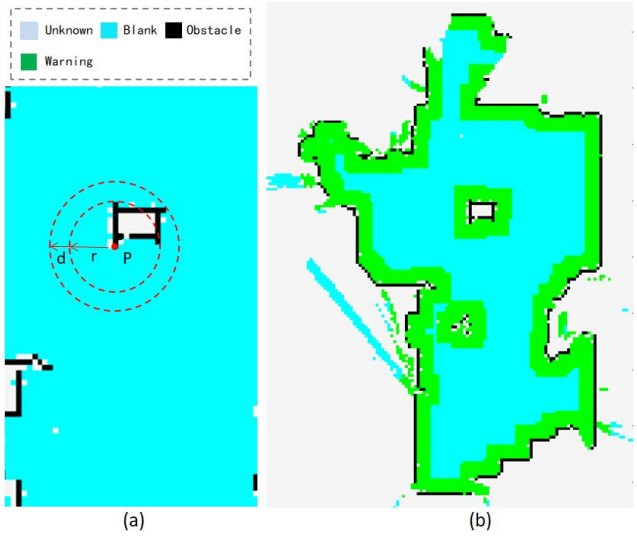

FIGURE 3. Warning area.

B. THE FLOW OF THE NEW A-STAR ALGORITHM

Driven by the cost function, the A* algorithm starts to search from the starting point until it finds the target point. The cost function f(n) is shown in Equation 1.

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

Among them, f(n) is the cost value of node P_n, g(n) is the actual path value of node P_n, and h(n) is the heuristic value.

The A-star algorithm based on the hybrid heuristic function includes the following steps.

Step 1: Put the starting point into the open-list, and put these blank grids adjacent to the starting point into the open-list.

Step 2: Take a node p(n) from the open-list as the current node. Calculate the cost function f(n+k) of its adjacent node p(n+k), where k=[1,2,…,8]. Calculate the difference between the cost function of the current node and the adjacent node, the calculation formula is

$$dif(k) = f(n+k) - f(n) \tag{2}$$

TABLE 1. Pseudo code of the new A-star algorithm.

```
algorithm Fun Astar( start, n, goal )
1
2   if IsReachGoal(start) == goal
3       return FindPath(start,goal)
4   end
5   open-list ← neibourPoint(start)
6   close-list ← 0
7
8   while open-list != 0
9       pn ← open-list(n)
10      if pn==goal
11          return FindPath(start,goal)
12      else
13          p(n+k) ← searchNeibors( pn )
14      end
15      for all p(n+k)
16          if p(n+k) ∉ blank
17              continue;
18          end
19          if p(n+k) ∈ open-list || p(n+k) ∈ close-list
20              continue;
21          end
22          dif(k) = f(n+k) – f(n)
23          if dif(k)>th
24              continue;
25          end
26          open-list ← p(n+k)
27      end
28      close-list ← p(n)
29  end

30  FindPath(start,goal)
```

Among them, f(n) is the cost value of the current node, and f(n+k) is the cost value of the k-th adjacent grid. Select free grids whose dif(k) is less than the threshold from the adjacent grids and add them to the open-list. Free grid refers to grids that are not added to the open-list and close-list. Remove the current node from the open-list and add it to the close-list.

Step 3: Record the current node as the parent node of the new nodes.

Step 4: Repeat 2-3 until the target point is added to the close-list.

Step 5: According to the relationship between the parent nodes and the child nodes, a path from the starting point to the target point is obtained.

The pseudocode of the A-star algorithm based on the hybrid heuristic function is shown in Table 1.

C. HYBRID HEURISTIC FUNCTION

The heuristic function of the A-star algorithm describes the distance between the node and the target point. Commonly used distance forms are Chebyshev distance, Manhattan distance, diagonal distance, and Euclidean distance.

The Chebyshev distance is shown in Equation 3.

$$h(n) = d \times \max(|x_n - x_N|, |y_n - y_N|) \tag{3}$$

In the formula, (x_n, y_n) is the coordinate of the node P_n, (x_N, y_N) is the coordinate of the target point P_N, and d is the weighting coefficient. Chebyshev calculates the x-direction distance and y-direction distance between the node and the target point and selects the larger absolute value as the output.

The Manhattan distance is shown in Equation 4.

$$h(n) = d \times (|x_n - x_N| + |y_n - y_N| \tag{4}$$

It calculates the x-direction distance and y-direction distance between the node and the target point and uses the sum of their absolute values as the node's heuristic value.

The diagonal distance is shown in Equation 5.

$$h(n) = d \times \sqrt{2} \times h_1(n) + d \times (h_2(n) - 2 \times h_1(n)) \tag{5}$$

Among them, $h_1(n) = d^* \min(|x_n - x_N|, |y_n - y_N|)$ is the shorter one the distance between the node and the target point in the x-direction and the y-direction. $h_2(n) = d^*(|x_n - x_N| + |y_n - y_N|)$ is the Manhattan distance of two points. h(n) is the diagonal distance, which roughly approximates the Euclidean distance, avoiding the squaring and exploiting operation.

The Euclidean distance is shown in Equation 6.

$$h(n) = d \times \text{sqrt}((x_n - x_N)^2 + (y_n - y_N)^2) \tag{6}$$

To reduce the number of search nodes of the A* algorithm in the indoor environment, a new hybrid heuristic function is proposed here, as shown in Equation 7.

$$h(n) = \frac{|x_n \times a + y_n \times b + c|}{\text{sqrt}(a^2 + b^2)} + \text{sqrt}\left((x_n - x_N)^2 + (y_n - y_N)^2\right) \tag{7}$$

Among them, $h_1(n) = |x_n{}^*a + y_n{}^*b + c| / \text{sqrt}(a^*a + b^*b)$ is the distance from the node $p_n(x_n, y_n)$ to the line $a^*x + b^*y + c = 0$. The line crosses the start point and the taget point. $h_2(n) = d^*\text{sqrt}((x_n - x_N)^2 + (y_n - y_N)^2)$ is the Euclidean distance from the node $p_n(x_n, y_n)$ to the target point $p_N(x_N, y_N)$.

Fig 4 shows the difference between the new heuristic function and the traditional heuristic function.

In the Fig4, the triangle is the starting point and the star is the target point. The blue part is the searched area. A is the reduced search area of the new algorithm compared to the old algorithm. The left image is the search effect of the traditional heuristic function. Here, the heuristic function of Euclidean distance is used as an example, which uses the Euclidean distance from the node to the target point as the driving condition to search. The right image is the search effect of the new heuristic function, where the red line is the line formed by the starting point and the target point. The new heuristic function is driven by the Euclidean distance from the node to the target point and the projected distance from the node to the line. As shown in the figure, for the same node,

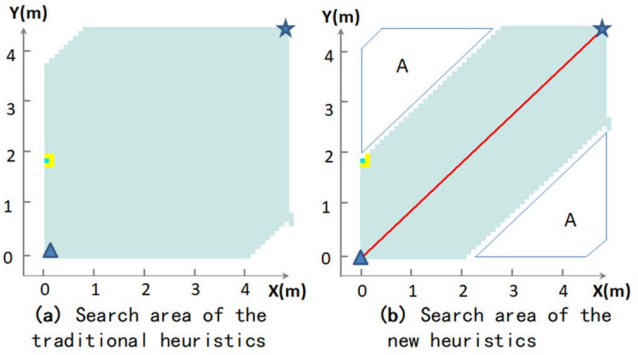

FIGURE 4. The new heuristic function and the traditional heuristic function.

the new heuristic function can approach the target point faster with fewer expansion directions, thereby reducing the number of nodes in the search area.

D. SELECTION OF EXPANSION THRESHOLDS

The A-star algorithm expands the path by analyzing the heuristic value of the current node and its adjacent nodes, and each node allows up to 8 search directions. The traditional A-star algorithm calculates the heuristic value of adjacent nodes and selects N nodes with the smallest heuristic value for expansion. For the same heuristic function, when N takes the maximum value, the A-star algorithm can ensure expansion to the target point, but at this time the search nodes are the most; the smaller the N is, the fewer the search nodes of the A-star algorithm, and the faster the search speed; but If N is too small, it is easy to cause the search process to be closed in advance so that the target point cannot be searched. Therefore, choosing an appropriate N value is a key to the A-star algorithm.

Different from the traditional method, this paper calculates the difference between the cost function of the current node and the adjacent nodes and selects the adjacent points whose difference is less than the expansion threshold to join the open-list. To this end, this paper builds the difference function as shown in Equation 2.

$$\mathrm{dif}(n,k) = \mathrm{f}(n+k) - \mathrm{f}(n)$$

Among them, f(n) is the cost value of the current node P(n), and f(n+k) is the cost value of the k-th adjacent node, $k \in [1, 8]$:

To choose an appropriate expansion threshold, the distribution of the solutions of the difference function is analyzed in this paper.

See Fig 5.a. $P_0(i_0, j_0)$ is the starting point. The dif(n,k) of 2000*2000 nodes around P_0 is calculated. Where n represents the n-th node and k represents the k-th direction. The coordinates of P_n is $P_n(i_0 + i_n, j_0 + j_n)$, $i_n \in [-1000, 1000]$, $j_n \in [-1000, 1000]$. The 8 differences of each P_n constitute a set of solutions.

See Fig 5.b. For the proposed hybrid heuristic function, the solution of dif(n,k) is related to the distance from the node to the line, which consists of the starting point and the target point. For this, we set 180 straight lines across the starting point.

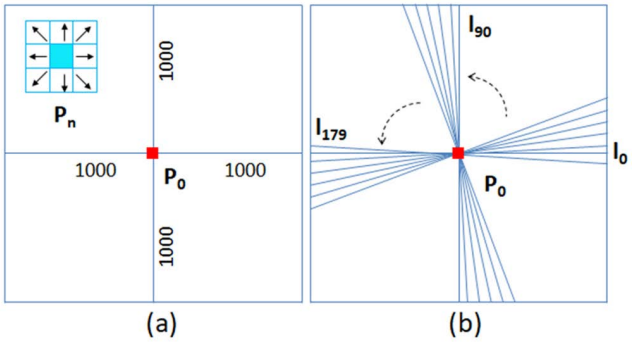

FIGURE 5. Range of nodes used to analyze dif(n, k).

For 4 traditional heuristic functions, each heuristic function obtains 2000*2000 = 4 million solutions. For the proposed hybrid heuristic function, 180*2000*2000 = 720 million solutions are obtained.

The above calculation process is implemented on Matlab7. The distribution of the solutions of the difference function is shown in Figure 6.

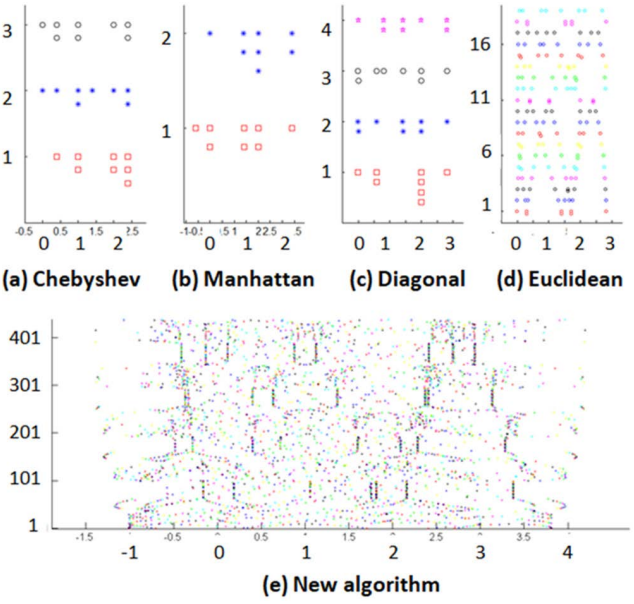

FIGURE 6. Distribution of solutions to dif(n, k).

In Figure 6, points of the same color constitute a set of solutions for the different functions. The vertical axis is the number of solutions, and the horizontal axis is the numerical value of the solutions.

As shown in the figure, the difference function of Chebyshev distance has 3 sets of different solutions, which are distributed on 6 discrete values. The difference function of Manhattan distance has 2 different sets of solutions, distributed over 5 discrete values. The difference function of the

diagonal distance has 4 different sets of solutions, distributed over 6 discrete values.

For the Euclidean distance and the proposed hybrid distance, the solution of the difference function is distributed over a continuous range of values. For the convenience of analysis, solutions with a difference within 0.1 are regarded as the same set of solutions. In this case, the difference function of Euclidean distance has 19 different sets of solutions, and the numerical values of the solutions are distributed between [0, 3]; the difference function of mixed distance has 440 sets of different solutions, and the numerical values of the solutions are Distributed between [−1.4, 4.2].

The influence of different expansion thresholds on the A-star algorithm is shown in Figure 7.

In the figure, the orange part is the expanded node, the black dot is the starting point, and the black square is the target point. For the same heuristic function, a large expansion threshold will make each node have more expansion directions, which increases the probability of the target point being searched, but also leads to the problem of too many search nodes. Conversely, a small augmentation threshold means fewer search directions, which will help reduce search nodes but increase the risk of not reaching the target point.

In the research and experiment process of this paper, to balance the search efficiency and success rate, the expansion threshold is set based on the standard that each node has at least 3 search directions. For the Chebyshev distance, the expansion threshold exp_th=1.8 is set. For Manhattan distance, set its expansion threshold exp_th=1.8. Diagonal distance, set its expansion threshold exp_th=1.8. For Euclidean distance, set its expansion threshold exp_th=1.5; for the proposed hybrid heuristic function, set its expansion threshold exp_th=2.3.

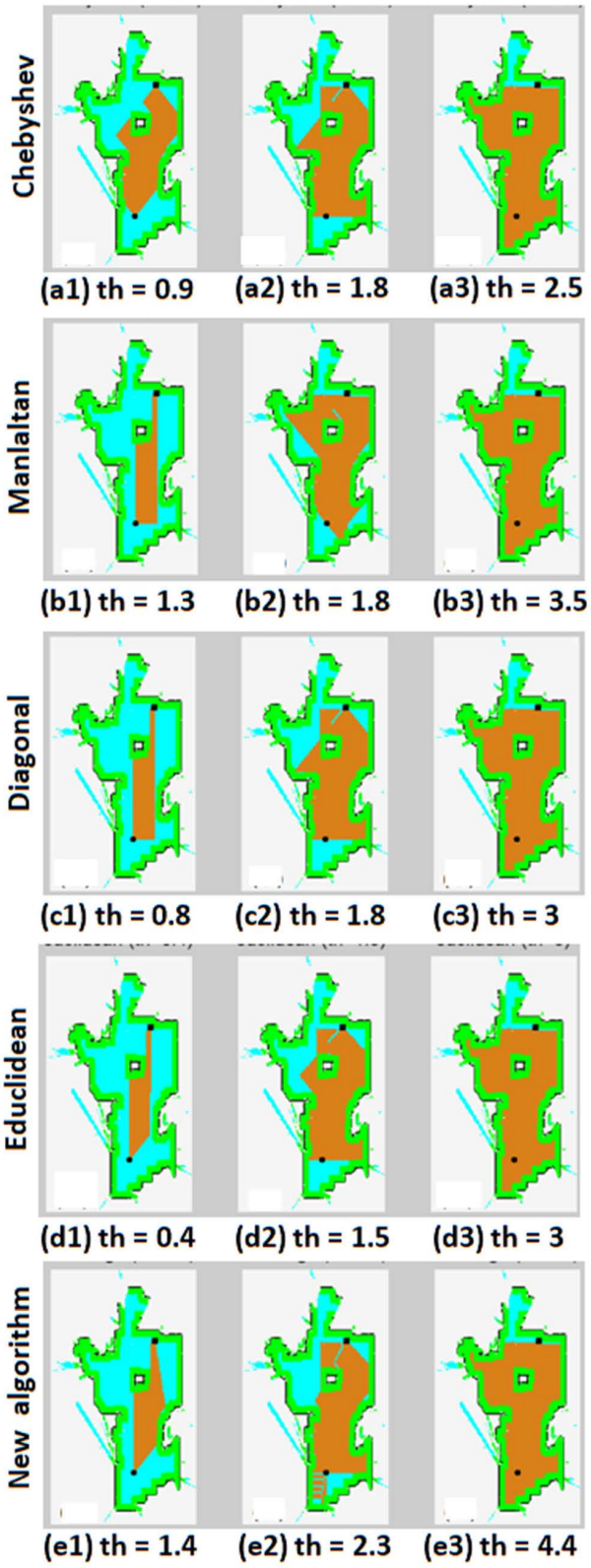

FIGURE 7. Influence of expansion threshold on A-star algorithm.

III. ASL-DWA: AN IMPROVED A-STAR ALGORITHM

The A-star algorithm obtains the global path from the starting point to the target point in the static map. There are polylines on the global path, which are not conducive to the smoothness of the cleaning robot's movement, and cannot guide the robot to avoid local unknown obstacles [8]. The DWA algorithm can obtain a path that conforms to the robot kinematics and can guide the robot to avoid local unknown obstacles, but it is easy to fall into the local optimum [10]. To enable the robot not only to be guided by global information, but also to avoid local unknown obstacles, a global yaw angle is designed here, and based on this, and improved A-star path planning method based on the global yaw angle is constructed.

A. KINEMATICS MODEL OF THE CLEANING ROBOT

The kinematic model of the cleaning robot is shown in Fig 8. The purple rectangle and the red rectangle are the driving wheels of the robot, and the blue triangles is the guiding wheel.

As shown in the figure, the robot is circular, and the outline radius is $R_1 = 17$cm. There are two drive wheels and one guide wheel. The distance between the drive wheel and the center of the robot is $R_2 = 12$cm, and the distance between the guide wheel and the center of the robot is $R_3 = 11.5$cm. In the figure, v is the linear velocity of the robot center, ω is

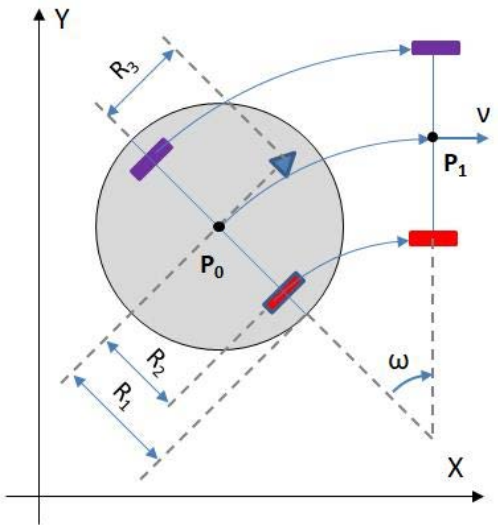

FIGURE 8. Kinematics model of the cleaning robot.

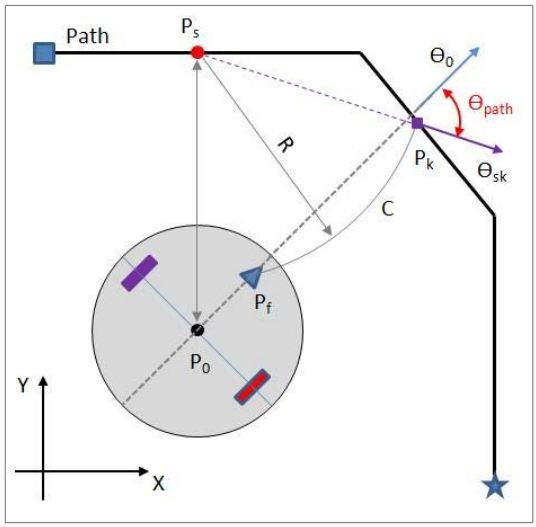

FIGURE 9. Global path yaw angle.

the angular velocity of the robot, p_0 is the pose of the robot at time t, and p_1 is the pose of the robot at time t+1. The kinematics model of the robot is shown in Equation 8.

$$\begin{cases} x_{t+1} = x_t - \frac{v}{\omega}\sin(\theta_t) + \frac{v}{\omega}\sin(\theta_t + \omega\Delta t) \\ y_{t+1} = y_t - \frac{v}{\omega}\cos(\theta_t) - \frac{v}{\omega}\cos(\theta_t + \omega\Delta t) \\ \theta_{t+1} = \theta_t + \omega\Delta t \end{cases} \tag{8}$$

Among them, Δt is the sampling interval, (x_t, y_t, θ_t) is the pose of the robot at time t, and $(x_{t+1}, y_{t+1}, \theta_{t+1})$ is the pose at time t + Δt.

B. GLOBAL PATH YAW ANGLE

1) PRINCIPLE

The proposed global path yaw angle characterizes the relative relationship between the robot's real-time attitude and the global path, as shown in Fig 9. The black line is the global path, the blue square is the starting point, the blue star is the target point, the red dot is the closest point, and the purple square is the forward point.

In the figure, Path is the global path, and $p_0(x_0, y_0, \theta_0)$ is the pose of the robot. $p_f(x_f, y_f)$ is the coordinate of the guide wheel. $P_s(x_s, y_s)$ is the closest point to the robot on the global path, which can be obtained from Equation 9.

$$\begin{cases} p_s \in \min(p_m - p_0), & 1 \le m \le M \\ p_m \in \text{Path}, & 1 \le m \le M \end{cases} \tag{9}$$

Among them, (p_m) is a point set consisting of a series of points uniformly distributed from the starting point to the target point on the global path.

$R = \|P_s, P_f\|$ is the distance from P_s to the center of the lead wheel p_f. C is an arc with P_s as the center and R as the radius. $P_k(x_k, y_k)$ is the intersection of C and Path, called the forward point, which can be obtained by Equation 10.

$$p_k = p_m, \quad m \in \{(\text{Path}_{s \le m \le m}\, p_m) \bigcap C\} \tag{10}$$

Among them, $C \in \{(x - x_s)^2 + (y - y_s)^2 = R^2\}$.

θ_{sk} is the orientation angle of the vector $\overrightarrow{p_s p_k}$, called the global path heading angle, which can be obtained by Equation 11.

The difference between the robot's orientation θ_0 and the global path's orientation θ_{sk} is the global path yaw angle, denoted as θ_{path}, which can be obtained by Equation 12.

$$\theta_{sk} = \begin{cases} \text{atan}(\frac{y_k - y_s}{x_k - x_s}), & x \ge 0 \\ \text{atan}(\frac{y_k - y_s}{x_k - x_s}) + 180, & x < 0, y \ge 0 \\ \text{atan}(\frac{y_k - y_s}{x_k - x_s}) - 180, & x < 0, y < 0 \end{cases} \tag{11}$$

$$\theta_{path} = \begin{cases} |\theta_0 - \theta_{sk}|, & |\theta_0 - \theta_{sk}| \le 180 \\ 360 - |\theta_0 - \theta_{sk}|, & \text{other} \end{cases} \tag{12}$$

2) DISCUSSION

Fig 10 shows the solution for the global path yaw angle for several different cases. In the figure, the black line is the global path, and the arrow at the end of the path indicates the forward direction of the path. The silver-gray circle is the robot, the blue triangle is the guide wheel of the robot, and the outer corners of it indicate the direction of the robot. The black dot represents the robot center p_0. The red square represents the robot center at the closest point P_s of the global path, and the orange square represents the forward point P_k.

Fig 10.a shows the most ideal situation, where the robot center is on the global path, and the robot is oriented in the same way as the global path. In this case, the point P_s (x_s, y_s) closest to the robot center on the global path coincides with the robot center $p_0(x_0, y_0, \theta_0)$, and the forward

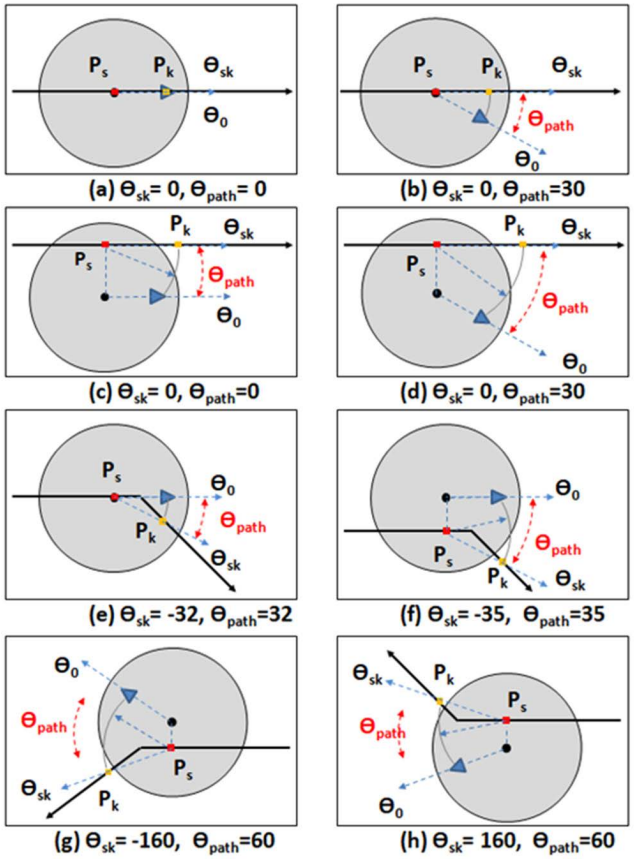

FIGURE 10. A few examples of global path yaw angle.

point $P_k(x_k, y_k)$ coincides with the robot guide wheel position $p_f(x_f, y_f)$. So, $\theta_{sk} = \theta_0$, global yaw angle $\theta_{path} = 0$.

In Fig 10.b, the robot center coincides with the global path, but the robot orientation is inconsistent with the global path. The forward direction of the global path is 0°. The orientation angle of the robot is $\theta_0 = -30°$. At this time, the point P_s (x_s, y_s) closest to the robot center on the global path coincides with the robot center $p_0(x_0, y_0, \theta_0)$. The forward point $P_k(x_k, y_k)$ is located on the global path in front of the robot. Therefore, θ_{sk}= 0, the global yaw angle $\theta_{path} = |\theta_0 - \theta_{sk}| = 30°$.

In Fig 10.c, the robot center is not on the global path, but the robot is facing the same direction as the global path. At this time, the point P_s (x_s, y_s) that is closest to the robot center on the global path is the vertical foot of the robot center on the global path, and the forward point $P_k(x_k, y_k)$ is located on the global path in front of the robot, and θ_{sk} is equal to the global path, so the global yaw angle $\theta_{path} = 0$.

In Fig 10.d, the robot's center is not on the global path, and the robot's orientation is also inconsistent with the global path's forward direction. The forward direction of the global path is 0°, and the robot heading angle is −30°. At this time, the point P_s (x_s, y_s) that is closest to the robot center on the global path is the vertical foot of the robot center on the global path, and the forward point $P_k(x_k, y_k)$ is located on the global path in front of the robot, so θ_{sk}= 0, the global yaw angle $\theta_{path} = |\theta_0 - \theta_{sk}| = 30°$.

In Fig 10.e, the robot is located near the turning point of the global path, the global path forward direction before turning is 0°, the global path forward direction after turning is −45°, and the robot's heading angle is 0°. At this time, the point P_s (x_s, y_s) closest to the robot center on the global path coincides with the robot center, and the forward point $P_k(x_k, y_k)$ is located on the global path after turning, so $\theta_{sk} = -32°$, the global yaw The angle $\theta_{path} = |\theta_0 - \theta_{sk}| = 32°$.

In Fig 10.f, the robot is located near the turning point of the global path, with a certain distance from both global paths. The forward direction of the global path before turning is 0°, the forward direction of the global path after turning is −45°, and the orientation angle of the robot is 0°. At this time, the point P_s (x_s, y_s) that is closest to the robot center on the global path is the vertical foot of the robot center on the global path before turning, and the forward point $P_k(x_k, y_k)$ is located on the global path after turning, so $\theta_{sk} = -35°$, global yaw angle $\theta_{path} = |\theta_0 - \theta_{sk}| = 35°$.

In Fig 10.g, $\theta_0 = 140°$, $\theta_{sk} = -160°$, global yaw angle $\theta_{path} = 360 - |\theta_0 - \theta_{sk}| = 60°$.

In Fig 10.h, $\theta_0 = -140°$, $\theta_{sk} = 160°$, global yaw angle $\theta_{path} = 360 - |\theta_0 - \theta_{sk}| = 60°$.

C. SCORING FUNCTION

The score function of the traditional DWA algorithm is shown in Equation 13.

$$\begin{aligned} G(v, \omega) = {} & \alpha \times \text{target\_heading}(v, \omega) \\ & + \beta \times \text{obs\_dist}(v, \omega) \\ & + \gamma \times \text{velocity}(v, \omega) \end{aligned} \tag{13}$$

In the formula, (v, ω) are the linear and angular velocities of the robot. (α, β, γ) are fixed weighting coefficients. *target_heading*(v, ω) is the score obtained by measuring the angle between the robot's orientation and the target's orientation, called the target orientation score: *obs_dist*(v,ω) is the score obtained by measuring the distance between the robot and the obstacle, which is called the obstacle distance score. *velocity*(v,ω) is the score obtained by measuring the difference between the robot's speed and the optimal speed, called the speed score.

The target orientation score can be obtained from Equation 14.

$$\text{target\_heading}(v, \omega) = 100 \times (1 - \frac{|\theta_0 - \theta_{tar}|}{180}) \tag{14}$$

Among them, θ_0 is the orientation angle of the robot, and θ_{tar} is the orientation angle of the robot to the target point.

The obstacle distance score can be obtained by Equation15.

$$\text{obs\_dist}(v, \omega) = \begin{cases} 0, & d_{obs} < d_{min} \\ 100, & d_{obs} > d_{max} \\ 100 \times \dfrac{d_{obs} - d_{min}}{d_{max} - d_{min}}, & \text{other} \end{cases} \tag{15}$$

Among them, d_{obs} is the shortest distance from the center of the robot to the obstacle. d_{min} is the minimum distance threshold, less than this threshold indicates that the robot is very close or has hit the obstacle. d_{max} is the maximum distance threshold, greater than this threshold indicates that the robot is far enough away from the obstacle. Within the threshold range, the larger the obstacle distance, the higher the score.

The speed score can be obtained by Equation 16.

$$\text{velocity}\,(v,\omega) = \gamma_v \times s_v + \gamma_\omega \times s_\omega \tag{16}$$

Among them, $(\gamma_v, \gamma_\omega)$ are scale factors, s_v can be obtained by Equation 17, and s_ω can be obtained by Equation 18.

$$s_v = \begin{cases} 0, & v < v_{min} \text{ or } v > v_{max} \\ 100 \times \left(1 - \dfrac{|v - v_0|}{v_{max} - v_{min}}\right), & \text{other} \end{cases} \tag{17}$$

Among them, v_{max} is the maximum linear velocity that the robot is allowed to achieve, v_{min} is the minimum linear velocity that the robot is allowed to achieve, and v_0 is the desired optimal linear velocity.

$$s_\omega = \begin{cases} 0, & \omega < \omega_{min} or \omega > \omega_{max} \\ 100 \times \left(1 - \dfrac{|\omega - \omega_0|}{\omega_{max} - \omega_{min}}\right), \\ \text{other} \end{cases} \tag{18}$$

Among them, ω_{max} is the maximum angular velocity that the robot is allowed to achieve, ω_{min} is the minimum angular velocity that the robot is allowed to achieve, and ω_0 is the desired optimal angular velocity.

The traditional A-star algorithm does not consider the kinematic characteristics of the robot, nor does it consider the real-time pose of the robot, the real-time distribution of obstacles, etc., so the obtained path is not conducive to the robot's execution, nor can it avoid unknown obstacles. Conversely, the traditional DWA score function only considers real-time information without the guidance of global information, so it is easy to cause the robot to fall into local optimum. A new scoring function is designed here, which combines the traditional scoring function with the global yaw angle, and introduces an adaptive weighting mechanism, thereby improving the A-star algorithm.

1) GLOBAL YAW SCORE

With the help of the global path yaw angle proposed in Section 3.2, the global yaw score is obtained, as shown in Equation 19, where θ_{path} is the global yaw angle, the larger the θ_{path}, the smaller the global yaw scores.

$$\text{path\_heading}\,(v,\omega) = 100 \times (1 - \frac{|\theta_{path}|}{180}) \tag{19}$$

2) GLOBAL PATH DISTANCE SCORE

The global path distance score is obtained according to the distance between the robot center and the global path, as shown in Equation 20.

$$\text{path\_dis}\,(v,\omega) = \begin{cases} 0, & dis > pd_{max} \\ 100, & dis < pd_{min} \\ 100 \times \dfrac{pd_{max} - dis}{pd_{max} - pd_{min}}, & \text{other} \end{cases} \tag{20}$$

3) SCORING FUNCTION WITH FIXED COEFFICIENTS

The global yaw score and the global path distance score are added to the traditional score function shown in Equation 13 to obtain a score function with a fixed coefficient, as shown in Equation 21.

$$\begin{aligned} G\,(v,\omega) = {} & \alpha \times \text{target\_heading}\,(v,\omega) \\ & + \beta \times \text{obs\_dist}\,(v,\omega) \\ & + \gamma \times \text{velocity}\,(v,\omega) \\ & + \tau \times \text{path\_heading}\,(v,\omega) \\ & + \delta \times \text{path\_dis}\,(v,\omega) \end{aligned} \tag{21}$$

4) SCORING FUNCTION WITH PREDICTION COEFFICIENTS

Equation 21 obtains the optimal speed to drive the robot to the next location. However, when the robot reaches the new location, it may find that it cannot proceed with the next path planning, that is, it is stuck in a local optimum. To reduce the probability of this phenomenon, the prediction coefficient is introduced in this paper, as shown in Equation 22.

$$\text{pre}(t,k) = \begin{cases} k, & \text{Min}_{i=1}^{i=t}(d_i) \le \text{pre}_{min} \\ 1, & \text{other} \end{cases} \tag{22}$$

Among them, t is the number of forecast periods and k is the decay coefficient. The robot moves at the current speed for t cycles, and the closest distance to the obstacle in each cycle is denoted as d_i. $\text{Min}_{i=1}^{i=t}(d_i)$ is the closest distance between the robot and the obstacle in t cycles. pre_{min} is the set minimum distance.

As shown in formula 22, the working principle of the prediction coefficient is: Assuming that the robot continues to move at the current speed, predict the closest distance between the robot and the obstacle in the next few cycles, if the distance is less than the threshold, multiply the score of the current speed by an attenuation factor k, thus reducing the probability of this speed is selected.

The score function with prediction coefficients is shown in Equation 23.

$$\begin{aligned} G\,(v,\omega) = {} & \{\alpha \times \text{target}_{\text{heading}(v,\omega)} \\ & + \beta \times \text{obs\_dist}\,(v,\omega) \\ & + \gamma \times \text{velocity}\,(v,\omega) \\ & + \tau \times \text{path\_heading}\,(v,\omega) \\ & + \delta \times \text{path\_dis}\,(v,\omega) \times \text{pre}(t,k) \end{aligned} \tag{23}$$

5) SCORING FUNCTION WITH ADAPTIVE COEFFICIENTS

When appropriate coefficients are selected, the score function shown in Equation 23 can drive the robot to approach the target point safely. However, when the robot arrives near the target point, the same coefficient may make it difficult for the robot to reach the target point accurately and quickly, especially when there are obstacles near the target point, or there are turning points in the global path near the target point. Likewise, another suitable set of coefficients can lead to good paths near the target point, whereas the same coefficients can lead to suboptimal paths farther from the target point.

To balance this conflict, this paper introduces an adaptive mechanism, which adjusts the weighting coefficient according to the change in the distance between the robot and the target point, and obtains an adaptive weighted score function, as shown in Equation 24.

$$\begin{aligned} G(v,\omega) = \{&\sigma(d) \times \alpha \times \text{target\_heading}(v,\omega) \\ &+ (1-\sigma(d)) \times \beta \times \text{obs\_dist}(v,\omega) \\ &+ (1-\sigma(d)) \times \gamma \times \text{velocity}(v,\omega) \\ &+ (1-\sigma(d)) \times \tau \times \text{pathheading}(v,\omega) \\ &+ (1-\sigma(d)) \times \delta \times \text{path\_dis}(v,\omega) \times \text{pre}(t,k) \end{aligned} \tag{24}$$

In the formula, $(\alpha, \beta, \gamma, \tau)$ is the set weighting coefficient, and σ(d) is the adaptive weighting coefficient, as shown in Equation 25, where d is the distance between the robot and the target point, and μ is the magnification.

$$\sigma(d) = 1/\exp(d \times \mu) \tag{25}$$

D. PARAMETER SETTING

1) SAMPLING PERIOD

The ASL-DWA algorithm makes a decision every once in a while, and this time interval is called the sampling period. At the start time t_i of the i-th sampling period, the algorithm calculates the scores for several groups of velocities (v_i, ω_i) in the sampling space by formula 24, and selects the combination (v, ω) with the highest score as the output, and drives the robot to move to the next A sampling time t_{i+1}.

Fig 11 shows the effect of different sampling periods on the path.

In the figure, the black dot is the starting point, the black square is the target point, and the red line is the path.

When other conditions are the same, the smaller the sampling period, the more sampling times, that is, the greater the amount of computation. At the same time, the too-small sampling period is also prone to produce a curved path, as shown by circle A in Fig. 11(a). Conversely, the larger the sampling period, the smaller the amount of computation. But too large a sampling period may make the path change too sluggish, as shown by circle B in Figure 11.c. At the same time, too large a sampling period may also cause trouble when the robot is about to reach the target point, as shown by circle C in Fig. 11(c).

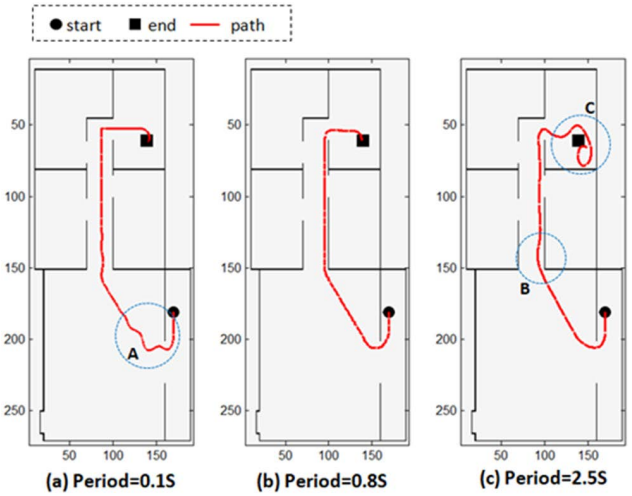

FIGURE 11. The effect of the sampling period on the path.

Considering the kinematic characteristics of the mobile robot, this paper selects T=0.8S. The ideal motion speed of the cleaning robot used is v=25cm/s, and the distance traveled in one sampling period is s=T∗v=20cm.

2) SAFE DISTANCE

The safety distance means that the robot maintains a sufficient distance from the obstacle so that the robot can avoid the obstacle in time. It should be noted that the mentioned obstacles refer to the obstacles in the forward direction of the robot.

Fig 12 shows the minimum distance dmin between the robot and the obstacle.

dmin can be obtained by Equation 26.

$$\begin{cases} \text{dmin} = \sqrt{R^2 - (R-2r)^2} \\ R(v/\omega) \times (180/\pi) \end{cases} \tag{26}$$

In the formula, r is the outer radius of the robot, v is the optimal linear velocity, and ω is the maximum angular velocity. For the cleaning robot used in this article: r=17cm, v=25cm/s, $\omega_{max} = 40°/s$. Calculation result: dmin=35.8cm.

3) PREDICTION PERIOD

As shown in Equation 22, the algorithm intervenes in the current speed score by predicting the closest distance between the robot and the obstacle in the next few cycles.

The effects of different forecast periods on path planning are shown in Figure 13. In the figure, the black dot is the starting point, the black square is the target point, the red dot is the moving obstacle, and the red line is the path.

See Figure 13.a, when the prediction period is 0 or too small, the algorithm may get stuck in a local optimum.

See Figure 13.c, when the prediction period is too large, the algorithm may get erroneous results at relatively narrow channels.

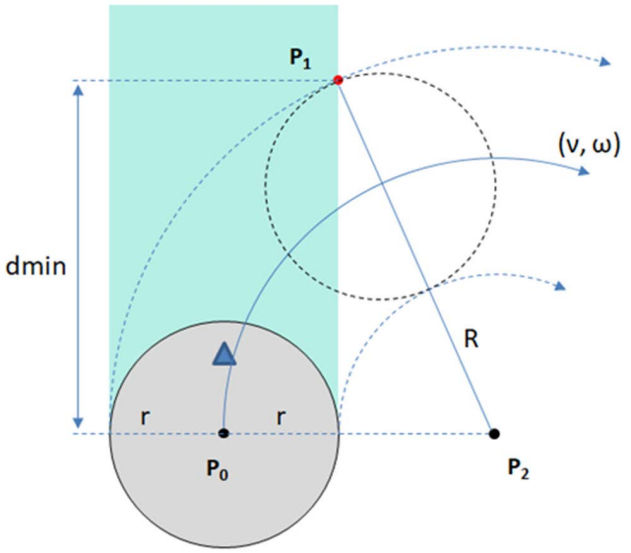

FIGURE 12. Safety distance between robot and obstacles.

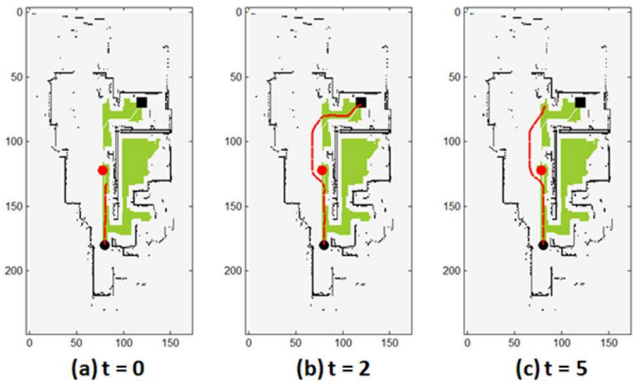

FIGURE 13. The effect of different Prediction periods on the path.

See Figure 13.b, a suitable prediction period can avoid the algorithm from falling into local optima.

4) SETTING OF σ(D)

As shown in Equation 24, this paper adaptively adjusts the weighting coefficients of the four scores according to the distance between the robot and the target point.

Fig 14 shows the effect of different σ(d) on path planning.

As shown in Figure 14.a, when the distance d is the same, the smaller the μ, the larger the σ(d), and the larger the coefficient of the target point heading score. At this time, the path may head to the target point prematurely.

As shown in Figure 14.c, the larger the μ, the smaller the σ(d), and the smaller the coefficient of the target point orientation score. At this time, it may be difficult for the path to reach the target point accurately.

As shown in Figure 14.b, a suitable value of μ can give ideal results. This paper takes σ(d)=1/exp(d∗0.12).

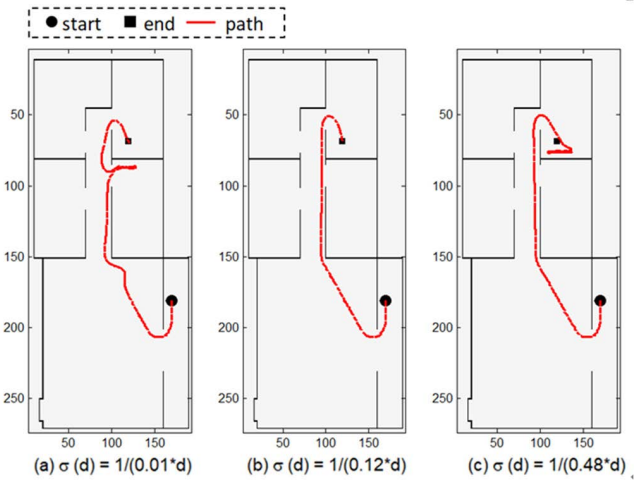

FIGURE 14. The effect of different σ(d) on the path.

E. DISCUSSION

1) IMPROVEMENT IN THE CASE WITHOUT UNKNOWN OBSTACLES

Figure 15 shows the improvement of the new algorithm on the A-star algorithm in the absence of unknown obstacles. In the figure, the gray area is the known obstacle; the black line is the path obtained by the A-star algorithm, and the arrow indicates the forward direction of the path; the purple line is the path obtained by the new algorithm.

Fig 15.a shows the most ideal situation where the robot is on the global path and the robot is oriented in the same direction as the global path is heading. In this case, the path obtained by the new algorithm is the same as the global path obtained by the A-star algorithm.

In Fig 15.b, the robot center coincides with the global path, but the robot orientation is not consistent with the global path. At this time, if the global path obtained by A star is followed, the robot needs to stop and rotate in the same direction as the global path. The path obtained by the adaptive weighted score function enables the robot to avoid pauses through arc motion and improve the action efficiency.

In Fig 15.c, the robot is located near the turning point of the global path. The forward direction of the global path before the turning is 0°, the forward direction of the global path after the turning is −45°, and the robot heading angle is 0°. If the global path obtained by A star is followed, the robot needs to stop and turn in the same direction as the global path. Driven by the global yaw angle, the new score function obtains a smooth path, which enables the robot to move more smoothly to the global path after turning.

2) IMPROVEMENT IN THE CASE WITH UNKNOWN OBSTACLE

The robot may encounter unknown obstacles during the movement, which do not appear on the static map. Such as obstructed and undetected obstacles, or changed positions of doors, furniture, people, etc. These unknown obstacles will cause some disturbance to path planning.

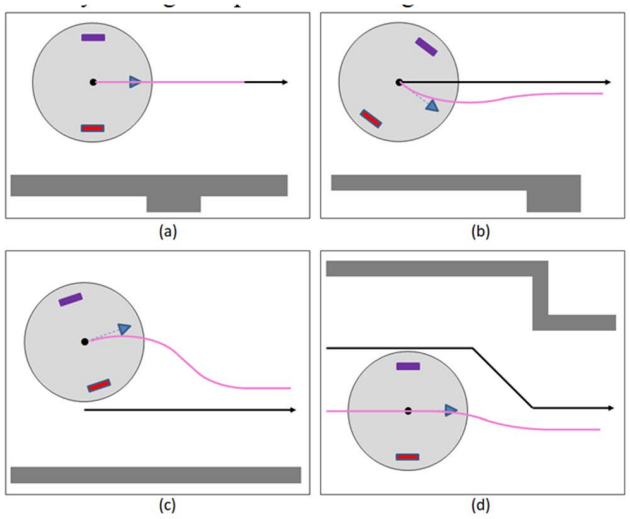

FIGURE 15. Improvement in the case without unknown obstacles.

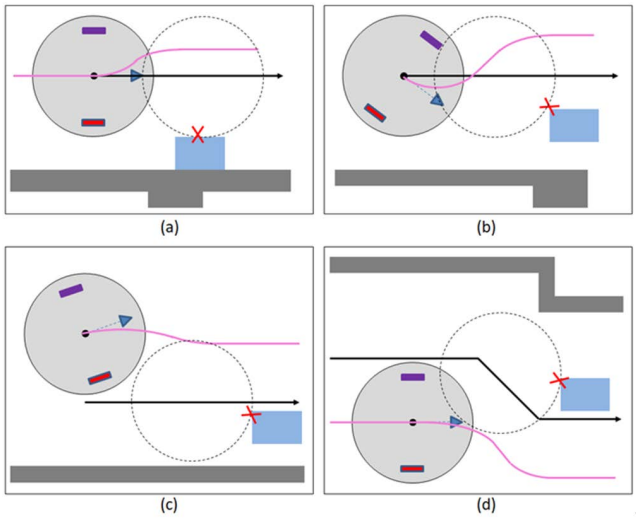

FIGURE 16. Improvement in the case with unknown obstacles.

Fig 16 shows the improvement of the new algorithm to the A-star algorithm in the presence of unknown obstacles. In the figure, the gray squares are known obstacles, and the blue squares are unknown obstacles; the black line is the path obtained by the A-star algorithm, and the arrow indicates the forward direction of the path; the red x shows that the path obtained by the traditional A∗ algorithm will be Unknown obstacles interrupt; the purple line is the path obtained by the new algorithm.

In the four cases shown in Fig. 16(a)–(d), if the robot follows the global path obtained by A star, the robot will collide with the obstacle. In contrast, ASL-DWA gets a path to avoid obstacles driven by the new score function, avoiding the failure of path planning due to the appearance of unknown obstacles.

3) COMPARISON OF ADAPTIVE COEFFICIENTS AND FIXED COEFFICIENTS

Fig 17 shows a comparison of adaptive weighting coefficients with fixed weighting coefficients.

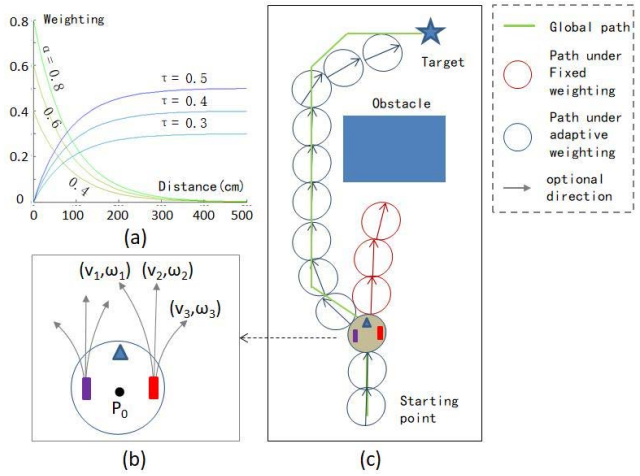

FIGURE 17. Comparison of adaptive coefficients and fixed coefficients.

In Fig 17.a the three curves marked with τ represent the relationship between the weighting coefficient $(1\text{-}\sigma(\mathrm{d}))*\tau$ and the distance corresponding to the global path yaw score when $\tau = 0.5$, 0.4, and 0.3, respectively. The three curves marked with α represent the relationship between the weighting coefficient $\sigma(\mathrm{d})*\alpha$ corresponding to the target orientation score and the distance when $\alpha = 0.8$, 0.6, and 0.4, respectively. As shown in the figure, when the distance between the robot and the target point is far, the weighting coefficient corresponding to the target orientation score is very small, while the weighting coefficient corresponding to the global path yaw score is close to τ. As the robot gradually approaches the target point, the weighting coefficient corresponding to the target orientation score gradually increases, while the weighting coefficient corresponding to the global path yaw score gradually decreases.

Figure 17.c shows the difference between the adaptively weighted scoring function (Equation 24) and the fixed coefficient scoring function (Equation 21) in planning paths under the same environment. At the node shown in Figure 16.b, the robot pose $\mathrm{p}_0\,(\mathrm{x}_0, \mathrm{y}_0, \theta_0) = (0, 0, 90)$, the linear velocity $\mathrm{v}_0 = 30\mathrm{cm/s}$, and the angular velocity $\omega_0 = 0°/\mathrm{s}$. Let $(\alpha, \beta, \gamma, \tau) = (0.4, 0.1, 0.1, 0.4)$, $\mu = 0.12$. In the speed sampling space, take 3-speed combinations as an example: $(\mathrm{v}_1, \omega_1) = (30, 40)$, $(\mathrm{v}_2, \omega_2) = (30, 0)$, $(\mathrm{v}_3, \omega_3) = (30, -40)$.

In the case of using a fixed coefficient score function, the scores corresponding to the 3-speed combinations are obtained by formula 21: $\mathrm{s}_1 = 82.1$, $\mathrm{s}_2 = 87.7$, $\mathrm{s}_3 = 78.0$. It can be seen that (v_2, ω_2) corresponds to the highest score, and the robot will execute at this speed. The final planned path is shown in the purple circle in Figure 17.c and the path fall into a local optimum.

In the case of using the adaptive weighted score function, the scores corresponding to the 3-speed combinations are obtained by formula 24: $\mathrm{s}_1 = 57.1$, $\mathrm{s}_2 = 50.0$, $\mathrm{s}_3 = 42.8$. It can be seen that (v_1, ω_1) corresponds to the highest score,

and the robot will execute at this speed. The final planned path is shown in the blue circle in Figure 17.c and the path reach the target smoothly.

It can be seen from the above analysis that although the global path information is added to the scoring function with fixed coefficients, it may still fall into the local optimum, while the scoring function using the adaptive weighting mechanism enables the robot to not only follow the guidance of the global path but also accurately and quickly. reach the target point.

4) COMPARISON OF NEW SCORING FUNCTION AND TRADITIONAL SCORING FUNCTION

Fig 18 shows a comparison of the adaptive weighted scoring function with the traditional scoring function.

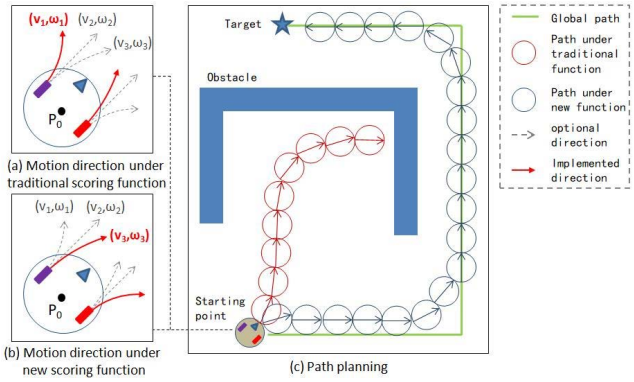

FIGURE 18. Comparison of new scoring function and Traditional scoring function.

As shown in Fig 18, at the starting point, the robot poses $p_0\,(x_0, y_0, \theta_0) = (0, 0, 45)$, the linear velocity $v_0 = 30\text{cm/s}$, and the angular velocity $\omega_0 = 0°/\text{s}$. In the conventional score function shown in Equation 13, let $(\sigma, \alpha, \beta, \gamma) = (1.0, 0.4, 0.4, 0.2)$. In the speed sampling space, take 3 speed combinations as example: $(v_1, \omega_1) = (30, 40)$, $(v_2, \omega_2) = (30, 0)$, $(v_3, \omega_3) = (30, -40)$. Their corresponding scores are obtained by formula 13: $s_1 = 98.2, s_2 = 91.1, s_3 = 83.5$. It can be seen that (v_1, ω_1) corresponds to the highest score, so the robot executes this speed, as shown by the red arrow in Figure 18.a, and the final planned path is shown in the purple circle in Figure 18.c.

Similarly, in the new score function shown in Equation 24, let $(\alpha, \beta, \gamma, \tau) = (0.4, 0.1, 0.1, 0.4)$, and let $\mu = 0.012$ in Equation 23. The scores corresponding to the 3-speed combinations are obtained by formula 24: $s_1 = 42.8$, $s_2 = 50.1, s_3 = 57.1$. It can be seen that (v_3, ω_3) corresponds to the highest score, so the robot executes this speed, as shown by the red arrow in Figure 18.b, and the final planned path is shown in the blue circle in Figure 18.c.

It can be seen from the above analysis that in the indoor environment where the distribution of obstacles is relatively dense, the traditional scoring function can easily lead to the machine falling into the local optimum and cannot reach the target, while the new scoring function can avoid falling into the local optimum under the guidance of the global path information.

F. ASL-DWA ALGORITHM FLOW

The proposed ASL-DWA algorithm flow is shown in Figure 19.

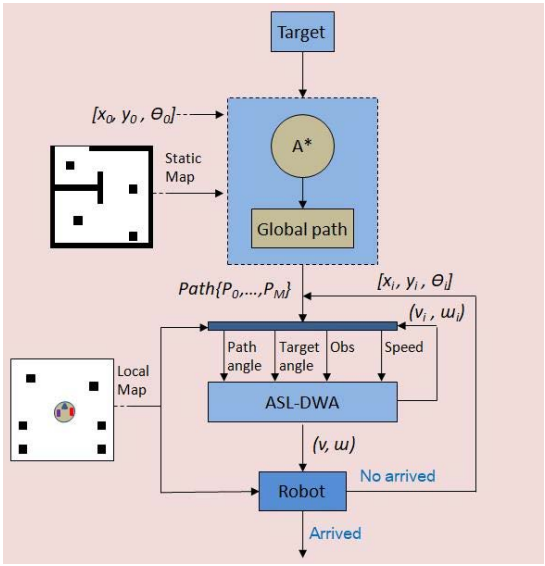

FIGURE 19. ASL-DWA algorithm flow.

The algorithm mainly includes the following steps:

(1) Run the A∗ algorithm based on the hybrid heuristic function. Within the global static map, search from the starting point until the target point is found. The input information at this stage mainly includes the current pose of the robot, the target point, and the global static map.

(2) Global path planning. According to the search result of A star, a global path from the out point to the target point is obtained, denoted as $Path\{p_0, \ldots, p_M\}$.

(3) Calculate the score corresponding to the sampling speed (v_i, ω_i). Firstly, the global path yaw score, target heading angle score, obstacle distance score, and speed score are calculated respectively, and then the final score is obtained according to the adaptive weighted score function.

(4) Run ASL-DWA. That is, traverse the entire sampling space to obtain the optimal speed combination (v, ω) as the output result.

(5) The robot runs one sampling period with velocity (v,ω).

In the above steps, 1 and 2 only need to be executed once, and 3-5 are executed in a loop until the target point is reached.

IV. EXPERIMENT AND RESULT ANALYSIS

A. COMPARISON OF HYBRID HEURISTIC FUNCTION AND TRADITIONAL HEURISTIC FUNCTION

To compare the pathfinding performance of hybrid heuristic functions and four traditional heuristic functions in indoor environments, we test five heuristic functions in three different environments.

The test environment is shown in Figure 20.

Fig 20.a1 shows test environment 1, which is a 5m∗14m office with 2 rooms, and forms a connected environment through doors. Obstacles such as desks, chairs, cartons, etc. exist in the environment. Fig 20.a2 shows test environment 2, which is a 12m∗10m apartment and consists of a living room, a bedroom, a kitchen, a bathroom, and a balcony, and there is a corridor between the room and the living room. Fig 20.a3 shows test environment 3, which is a home environment with a living room, two rooms, a kitchen, a bathroom, and two balconies.

Figure 20.b1-b3 are the grid maps obtained by the cleaning robot in the three environments. Each grid in the figure corresponds to an area of 5cm∗5cm. The black part is the obstacle, the cyan part is the passable area, and the silver part is the unknown area. The green part is the early warning area, that is, although there are no obstacles in this area, the center of the machine cannot reach these areas due to the limitation of the robot's overall size and safety distance.

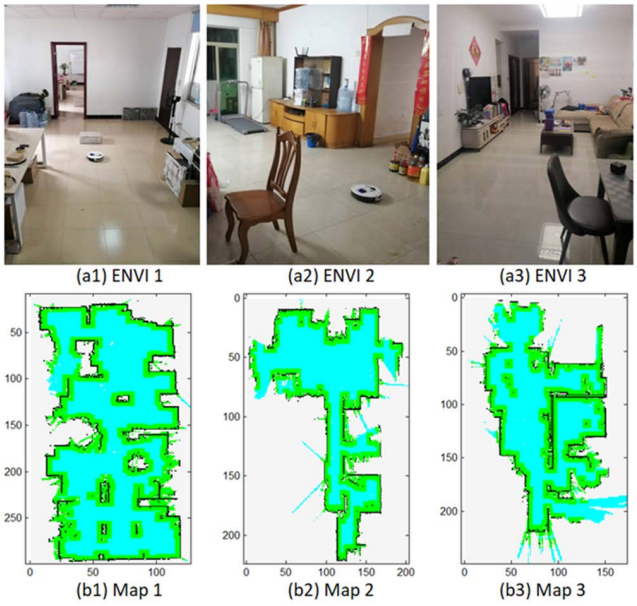

FIGURE 20. Test environments and maps.

There are some incomplete detection parts in the map, this is due to a) There is a height difference between the ground of the kitchen, balcony, bathroom, and other environments and the ground of other rooms, the cleaning robot will not enter these areas to avoid being trapped; b) There are areas that won't completely block the laser detection signal, but the robot can't pass through, such as areas cut off by chairs, or narrow passages between beds and cabinets.

We test the heuristic function in two different situations: the case where the starting point and the target point are in the same room and the case where the starting point and the target point are in different rooms.

1) THE STARTING POINT AND THE TARGET POINT ARE IN THE SAME ROOM

Fig 21 shows the search results of five heuristic functions in three maps when the starting point and the target point are in the same room. In the figure, the black dot is the starting point, the black square is the target point, and the orange area is the node to which the algorithm is extended.

As shown in Figure 21, when the starting point and the target point are in the same room, there are relatively few obstacles between the two points, and the five heuristic functions can reach the target point with a relatively small search area.

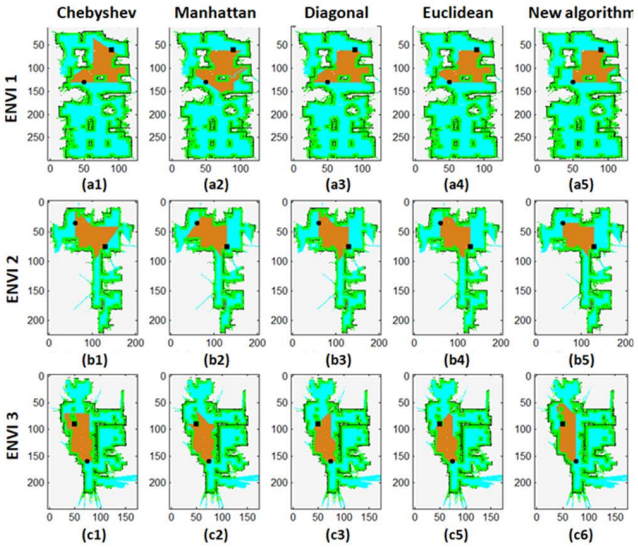

FIGURE 21. Search results for five heuristics (points in the same room).

The number of search nodes for the five heuristic functions is shown in Table 2.

TABLE 2. The number of search nodes (points in the same room).

	chebyshev	manhattan	diagonal	euclidean	new
ENV 1	3226	3431	2891	2787	2414
ENV 2	3691	3342	3144	3048	2713
ENV 3	3178	2693	2570	2586	2649

As shown in Table 2, compared with the heuristic function based on Chebyshev distance, the proposed hybrid heuristic function reduces the number of nodes in the three maps by 25.17%, 26.5%, and 16.6% respectively, with an average reduction of 22.7%. Compared with the heuristic function based on Manhattan distance, the number of nodes is reduced by 29.6%, 18.8%, and 1.6%, and the average is 16.7; compared with the heuristic function based on diagonal distance, the number of nodes is reduced by 16.5%, 13.7 %, -3%, an average reduction of 9%; compared with the heuristic function based on Euclidean distance, the number of nodes

is reduced by 13.4%, 11%, -2.4%, and an average reduction of 7.3%.

2) THE STARTING POINT AND THE TARGET POINT ARE IN DIFFERENT ROOMS

Fig 22 shows the search results of five heuristic functions in three maps when the starting point and the target point are in different rooms.

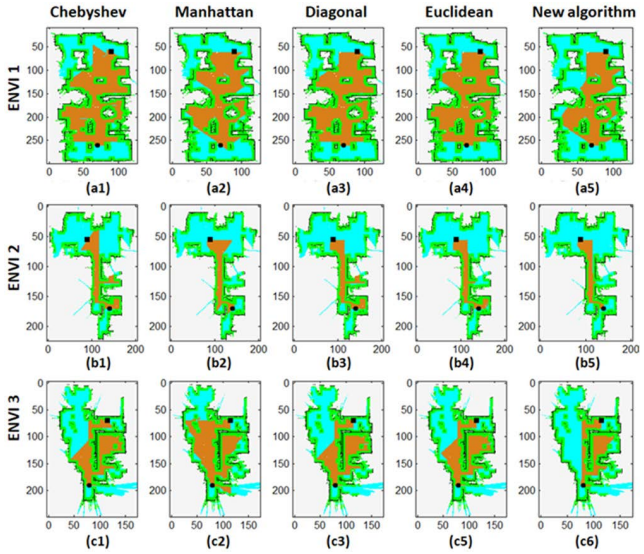

FIGURE 22. Search results for five heuristics (points in different rooms).

As shown in Figure 22, when the starting point and the target point are in different rooms, there are relatively many obstacles between the two points, and the heuristic function needs a large search area to reach the target point.

Table 3 shows the number of search nodes for the five heuristic functions in the three grid maps.

TABLE 3. The number of search nodes (points in different rooms).

	chebyshev	manhattan	diagonal	euclidean	new
ENV 1	8464	7520	8368	8285	6308
ENV 2	2710	2267	2343	2088	1712
ENV 3	4056	5736	4000	4083	2744

As shown in Table 3, compared with the heuristic function based on the Kichebyshev distance, the proposed hybrid heuristic function reduces the number of search nodes in the three maps by 25.5%, 36.8%, and 32.4% respectively, with an average reduction of 31.5%; Compared with the heuristic function based on the distance, the search nodes are reduced by 16.1%, 24.5%, and 52.2%, respectively, with an average reduction of 30.9%; compared with the heuristic function based on the diagonal distance, the search nodes are reduced by 24.6%, 26.9%, and 31.4% %, an average reduction of 27.6%; compared with the Euclidean distance-based heuristic function, the search nodes are reduced by 23.9%, 18%, and 32.8%, respectively, with an average reduction of 24.9%.

B. COMPARISON OF ASL-DWA AND TRADITIONAL A-STAR ALGORITHM

We use several indicators to quantify the pathfinding effect of the algorithm: the number of static times, the rotation angle in situ, the path length, the movement time, and whether the target is reached.

Static times refer to the number of times the robot stops moving, which is recorded as N_{static}.

The in-situ rotation angle refers to the rotation angle of the robot around the center of the machine, denoted as A_{turn}.

The path length refers to the movement distance of the robot from the starting point to the target point, denoted as L_{path}. To simplify the calculation, we store the path as N discrete points at certain distance intervals. The path length can be obtained by Equation 27.

$$L_{path} = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{27}$$

Among them, N is the number of points, and (x_i, y_i) is the coordinates of the i-th point on the path.

Movement time refers to the time taken by the robot to move from the starting point to the target point, which is recorded as T_{move}. Its calculation method is shown in Equation 28.

$$T_{move} = \sum_{i=1}^{M} s_i/v_i + N_{static} \times t_{static} + A_{turn}/\omega_{turn} \tag{28}$$

Among them, s_i is the arc length of the robot in the i-th sampling period, v_i is the linear velocity of the robot in the i-th sampling period, and M is the number of sampling periods. For the traditional A∗ algorithm, the linear speed of the robot is v=25cm/s. For the ASL-DWA algorithm, the optimal linear speed of the robot is also 25cm/s, but the actual movement speed is determined by the real-time calculation results. N_{static} is the number of times the robot stops, and t_{static} is the time for each stop. Taking into account the braking time, the elimination of inertia time, and other factors, the stopping time of the robot is $t_{static} = 500$ms. A_{turn} is the angle which the robot turns around its center. ω_{turn} is the angular velocity when rotating in place, and the angular velocity of the robot rotating in place is $\omega = 40°/S$.

To compare the pathfinding effect of the ASL-DWA algorithm and the traditional A-star algorithm, we tested both algorithms in the three environments shown in Figure 20.

1) THE CASE WITHOUT UNKNOWN OBSTACLES

In the absence of unknown obstacles, the real-time map during robot movement is the same as the static map used in A-star path planning. The path planning results of the traditional A-star algorithm and the proposed ASL-DWA algorithm in three environments are shown in Figure 23.

In the figure, the black dot is the starting point, the black square is the target point, the blue line is the path obtained by

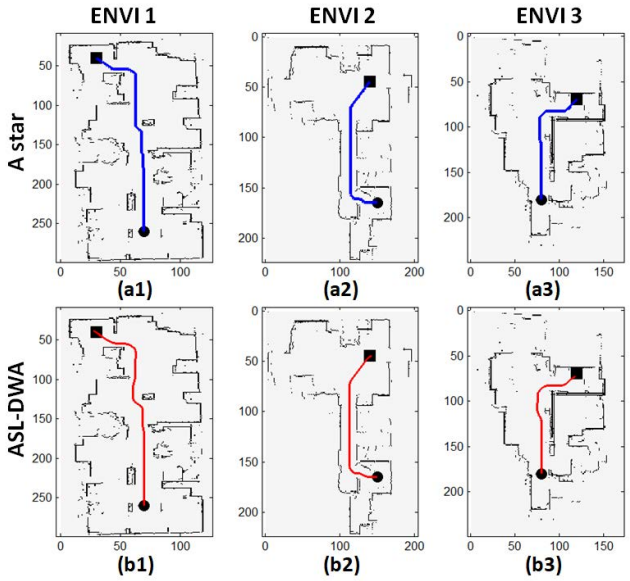

FIGURE 23. Comparison of ASL-DWA and traditional A star (without unknown obstacles).

the traditional A-star algorithm, and the red line is the path obtained by the ASL-DWA algorithm.

As shown in the figure, both algorithms can reach the target point, but the path of the traditional A-star algorithm has many polylines, while the path of ASL-DWA is smoother and avoids the polyline path.

The parameters of the paths obtained by the two algorithms are shown in Table 4.

TABLE 4. Comparison of ASL-DWA and traditional A star (without unknown obstacles).

	A star					ASL-DWA				
	N_{static}	A_{turn}	L_{path}	T_{move}	Target Arrived	N_{static}	A_{turn}	L_{path}	T_{move}	Target Arrived
ENV 1	11	495°	1221cm	66.7s	yes	0	0°	1220cm	40.8s	yes
ENV 2	7	315°	803.3cm	43.5s	yes	0	0°	802.7cm	27.2s	yes
ENV 3	5	225°	705.6cm	36.3s	yes	0	0°	705.2cm	23.2s	yes

As shown in Table 4, there are many polylines in the path obtained by the A-star algorithm, and the robot needs to stop at these places, and then rotate around the robot center at a certain angle, and then continue to move along the next path. In contrast, ASL-DWA avoids the stationary and in-situ rotation of the robot, thereby reducing the action time. ASL-DWA reduces the movement time of the robot in the three maps by 38.8%, 37.4%, and 36.1%, respectively.

2) THE CASE WITH UNKNOWN OBSTACLES

In the case of unknown obstacles, the path planning results of the proposed ASL-DWA algorithm and other algorithms are shown in Figure 24.

In the figure, the black dots represent the starting point, the black squares represent the target points, and the red dots represent unknown obstacles. The red line is the walking path

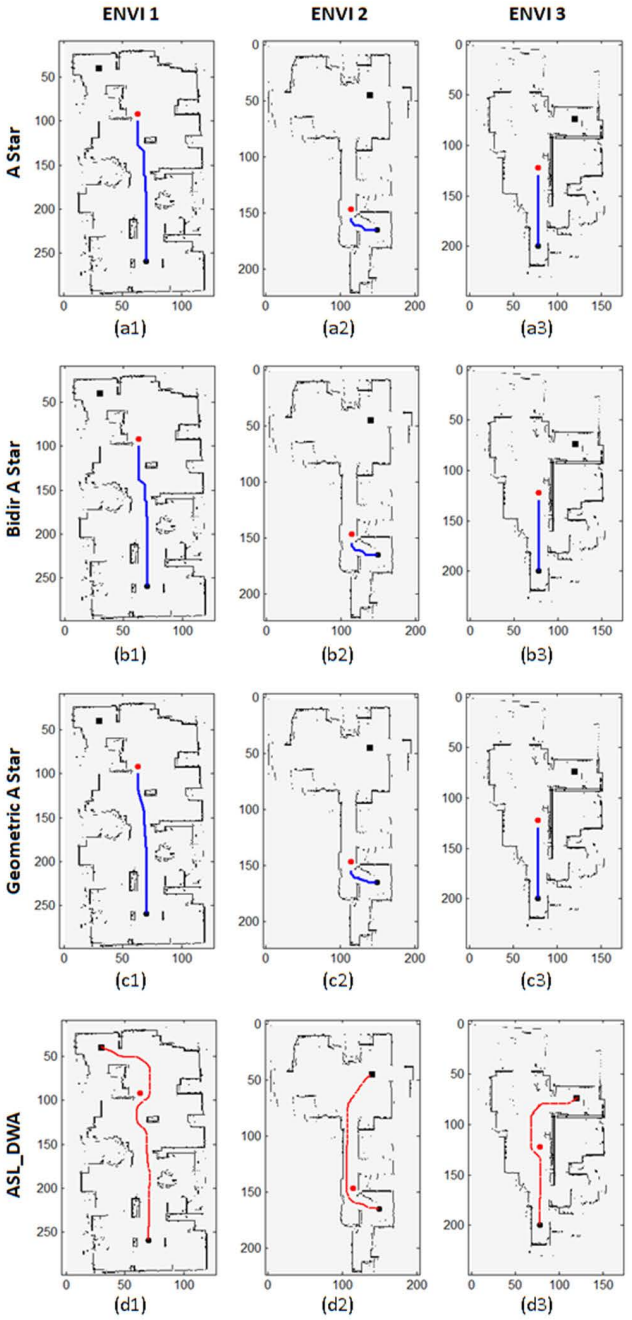

FIGURE 24. Comparison of new algorithm and other algorithms (with unknown obstacles).

of the mobile robot when using ASL-DWA and the blue line is the walking path using other algorithms.

As shown in Fig 24.a-c, when there are unknown obstacles near the global path, traditional algorithms cannot guide the robot to avoid these obstacles, resulting in an early stop of the path planning.

As shown in Fig 24.d, when the ASL-DWA algorithm is adopted, the robot can avoid moving obstacles and reach the target smoothly.

The parameters of the paths obtained by the two algorithms are shown in Table 5.

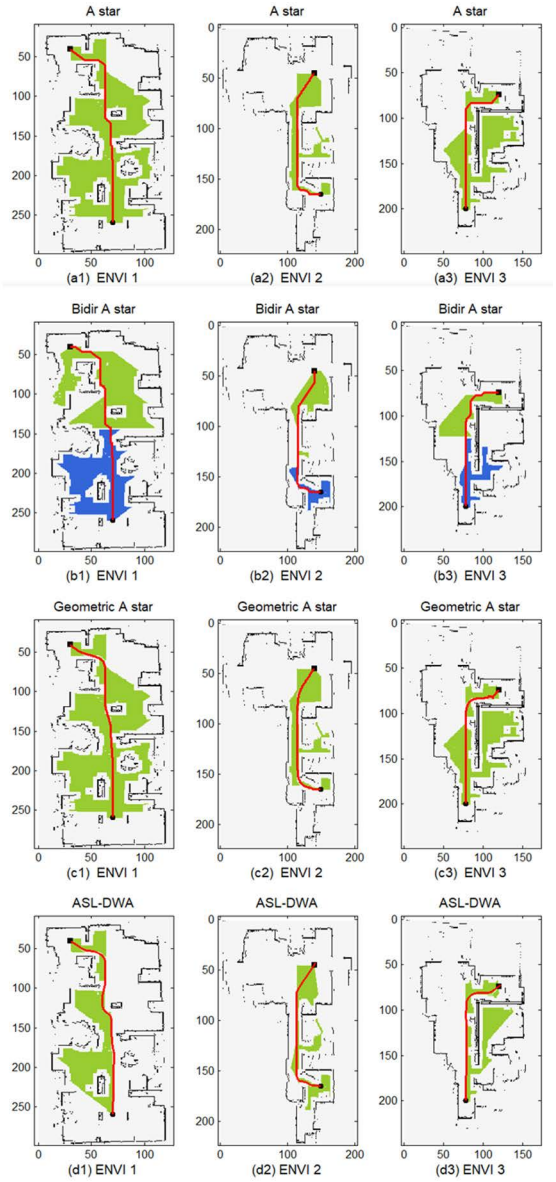

FIGURE 25. Comparison of ASL-DWA and other algorithms.

C. COMPARISON OF ASL-DWA WITH OTHER METHODS

To illustrate the effectiveness of ASL-DWA, we compare ASL-DWA with several other algorithms, including A-star, Bidirectional A-star [11], and Geometric A-star [18].

The path planning results of several algorithms in three environments are shown in Figure 25. In the figure, the green and blue areas are the nodes searched by the algorithm, and the red line is the path obtained by the algorithm.

Table 6 shows the parameters of the paths obtained by several algorithms.

As shown in Table 6, the proposed ASL-DWA algorithm reduces the number of nodes searched in all three environments and reduces the movement time of the machine from the starting point to the target point. Further, ASL-DWA can guide the machine to avoid unknown obstacles, while the other three algorithms cannot avoid unknown obstacles.

TABLE 5. Comparison of ASL-DWA and other algorithms (with unknown obstacles).

	Algorithm	N_{static}	A_{turn}	L_{path}	T_{move}	Arrived
ENV 1	A star	8	360°	814.5cm	/	No
	Bid A star	6	270°	814.5cm	/	No
	Geometric	0	0°	802.7cm	/	No
	ASL-DWA	0	0°	1219.8cm	42.4s	yes
ENV 2	A star	6	270°	201.6cm	/	No
	Bid A star	6	270°	201.6cm	/	No
	Geometric	0	0°	198.6cm	/	No
	ASL-DWA	0	0°	802.7cm	28s	yes
ENV 3	A star	0	0°	350cm	/	No
	Bid A star	0	0°	350cm	/	No
	Geometric	0	0°	350cm	/	No
	ASL-DWA	0	0°	792.8cm	28.8s	yes

TABLE 6. Comparison of ASL-DWA and other algorithms.

		Node	N_{static}	A_{turn}/°	L_{path}/cm	T_{move}/s	Unknown Obstacle Avoidence
ENV 1	A star	8337	11	495	1221	66.7	No
	Bid A star	8179	14	630	1221	71.6	No
	Geometric	8337	0	0	1189	47.6	No
	ASL-DWA	5687	0	0	1220	40	Yes
ENV 2	A star	2824	7	315	803.3	43.5	No
	Bid A star	2445	12	540	803.3	51.6	No
	Geometric	2824	0	0	778.7	31.1	No
	ASL-DWA	2459	0	0	802.7	27.2	Yes
ENV 3	A star	4040	3	135	793.1	36.6	No
	Bid A star	2782	8	360	793.1	44.7	No
	Geometric	4040	0	0	782.9	31.3	No
	ASL-DWA	2404	0	0	792.8	26.4	Yes

V. CONCLUSION

This paper introduces a new path planning algorithm called the ASL-DWA algorithm. It is suitable for indoor cleaning robots, which can reduce the number of search nodes of the traditional A∗ algorithm, eliminate polyline paths, guide robots to avoid unknown obstacles, and avoid falling into local optimum. Experiments show that the proposed ASL-DWA algorithm has obvious advantages compared with other algorithms.

There are several directions for future work. (1) In the indoor environment, not all of the ground is on one plane, and the path planning based on 3D sensor information is a direction worthy of research. (2) The working goal of the cleaning robot is to achieve full coverage of the working environment. How to combine the coverage algorithm with the path planning algorithm will be an interesting topic; (3) Multi-machine collaboration can make the cleaning robot work more efficiently, Research in this area is a hot topic.

REFERENCES

[1] K. Koide, J. Miura, M. Yokozuka, S. Oishi, and A. Banno, "Interactive 3D graph SLAM for map correction," *IEEE Robot. Autom. Lett.*, vol. 6, no. 1, pp. 40–47, Jan. 2021.

[2] K. Koide, J. Miura, and E. Menegatti, "A portable three-dimensional LIDAR-based system for long-term and wide-area people behavior measurement," *Int. J. Adv. Robotic Syst.*, vol. 16, no. 2, Mar. 2019, Art. no. 172988141984153.
[3] D. Teso-Fz-Betoño, E. Zulueta, U. Fernandez-Gamiz, I. Aramendia, and I. Uriarte, "A free navigation of an AGV to a non-static target with obstacle avoidance," *Electronics*, vol. 8, no. 2, p. 159, 2019.
[4] B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai, and H. Pan, "An improved A* algorithm for the industrial robot path planning with high success rate and short length," *Robot. Auto. Syst.*, vol. 106, pp. 26–37, Aug. 2018.
[5] D. Connell and H. Manh La, "Extended rapidly exploring random tree–based dynamic path planning and replanning for mobile robots," *Int. J. Adv. Robotic Syst.*, vol. 15, no. 3, May 2018, Art. no. 172988141877387.
[6] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Jan. 2019.
[7] W. Yin and X. Yang, "A totally astar-based multi-path algorithm for the recognition of reasonable route sets in vehicle navigation systems," *Proc. Social Behav. Sci.*, vol. 96, pp. 1069–1078, Nov. 2013.
[8] J. C. Mohanta, D. R. Parhi, and S. K. Patel, "Path planning strategy for autonomous mobile robot navigation using Petri-GA optimisation," *Comput. Electr. Eng.*, vol. 37, no. 6, pp. 1058–1070, Nov. 2011.
[9] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.
[10] F. H. Ajeil, I. K. Ibraheem, M. A. Sahib, and A. J. Humaidi, "Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106076.
[11] X. Wu, L. Xu, R. Zhen, and X. Wu, "Bi-directional adaptive A* algorithm toward optimal path planning for large-scale UAV under multi-constraints," *IEEE Access*, vol. 8, pp. 85431–85440, 2020.
[12] X. Miao, J. Lee, and B.-Y. Kang, "Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments," *IEEE Access*, vol. 6, pp. 38200–38215, 2018.
[13] Y.-C. Wang and K.-C. Chen, "Efficient path planning for a mobile sink to reliably gather data from sensors with diverse sensing rates and limited buffers," *IEEE Trans. Mobile Comput.*, vol. 18, no. 7, pp. 1527–1540, Jul. 2019.
[14] B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai, and H. Pan, "An improved A* algorithm for the industrial robot path planning with high success rate and short length," *Robot. Auto. Syst.*, vol. 106, pp. 26–37, Aug. 2018.
[15] Y. Zhang and L. Li, "Development of path planning approach using improved A-star algorithm in AGV system," *J. Internet Technol.*, vol. 20, no. 3, pp. 915–924, 2019.
[16] R. Song, Y. Liu, and R. Bucknall, "Smoothed A* algorithm for practical unmanned surface vehicle path planning," *Appl. Oceans Res.*, vol. 83, pp. 9–20, Feb. 2019.
[17] Z. Liu, H. Liu, Z. Lu, and Q. Zeng, "A dynamic fusion pathfinding algorithm using Delaunay triangulation and improved A-star for mobile robots," *IEEE Access*, vol. 9, pp. 20602–20621, 2021.
[18] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A-star algorithm: An improved A-star algorithm for AGV path planning in a port environment," *IEEE Access*, vol. 9, pp. 59196–59210, 2021.
[19] K. Mi, J. Zheng, Y. Wang, and J. Hu, "A multi-heuristic A* algorithm based on stagnation detection for path planning of manipulators in cluttered environments," *IEEE Access*, vol. 7, pp. 135870–135881, 2019.
[20] M. Elhoseny, A. Tharwat, and A. E. Hassanien, "Bezier curve based path planning in a dynamic field using modified genetic algorithm," *J. Comput. Sci.*, vol. 25, pp. 339–350, Mar. 2018.
[21] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.
[22] R. Kala, A. Shukla, and R. Tiwari, "Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning," *Artif. Intell. Rev.*, vol. 33, no. 4, pp. 307–327, Apr. 2010.
[23] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT*: An improved path planning algorithm for mobile robots," *Expert Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113425.
[24] C. Rosmann, W. Feiten, and T. Wosch, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. ROBOTIK 7th German Conf. Robotics*, May 2012, pp. 1–6.
[25] M. Islam, M. Okasha, and E. Sulaeman, "A model predictive control (MPC) approach on unit quaternion orientation based quadrotor for trajectory tracking," *Int. J. Control, Autom. Syst.*, vol. 17, no. 11, pp. 2819–2832, Nov. 2019.

HAOXIN LIU is with the School of Information and Communication Engineering, Hainan University, Haikou, China. His research interests include mobile robot SLAM algorithm, navigation algorithm, and motion control.

YONGHUI ZHANG is with the School of Information and Communication Engineering, Hainan University, Haikou, China. His research interests include embedded systems and intelligent detection.