Research on Dynamic Obstacle Avoidance Method of Mobile Robot based on Improved A* Algorithm and Dynamic Window Method

Zhangfang Hu, Xingyuan Wang, Junhao Zhang

Abstract—**In response to the imperative challenges associated with global path planning and dynamic obstacle avoidance for mobile robots navigating dynamic environments, we introduce a hybrid path planning methodology that synergistically combines an enhanced A* algorithm with an optimized dynamic window method. In the refined A* algorithm, an adaptive weight heuristic search function that comprehensively considers Manhattan distance and Euclidean distance is designed to improve the search efficiency; secondly, a redundant point elimination method is proposed to delete redundant paths node and perform path pruning, and then use Minimum Snap to smooth and optimize the pruned path. To address challenges associated with both random obstacle avoidance and dynamic obstacle avoidance, the fusion algorithm delineated in this study incorporates the global path nodes, derived through enhancements to the A* algorithm, as local target points while also employing the optimized dynamic window method for local path planning. The experimental results indicate that, on average, the improved A* algorithm can decrease the path length by 17.2% and reduce the number of search nodes by 62.3% compared to the conventional A* algorithm. After integrating and optimizing the dynamic window method, it can realize random obstacle avoidance and dynamic Avoidance.**

Index Terms—**Dynamic obstacle avoidance, Heuristic function, Dynamic window method, Hybrid path planning**

I. Introduction

Path planning stands as a pivotal technology in the research of autonomous robots, exerting a vital influence in the domain of mobile robot navigation. Path planning is categorized according to the robot's knowledge of map information, distinguishing between global path planning, contingent upon pre-existing global data [1-2], and local path planning, predicated on dynamic local information [3-4].

The A* algorithm [5-7] stands as a widely employed graph search method for addressing global optimal pathfinding in static environments. Integrating heuristic search with the Dijkstra algorithm, it boasts advantages such as concise path planning and computational simplicity. Nevertheless, traditional A* algorithms exhibit limitations, including elevated time complexity, suboptimal path smoothness, the presence of redundant nodes, and a tendency to closely approach obstacles. In light of these deficiencies, ŠišLák et al. introduced the Accelerated A* algorithm [8], aiming to enhance search efficiency while preserving optimality. This algorithm employs a heuristic distance estimation method to mitigate the computational complexity of each node, expediting the path planning process. However, it lacks consideration for safe distances from obstacles. Liu et al. [9] explored into the impact of the parent node of the current node on the search path, integrating information from the parent node into the heuristic function, and adjusting its weight to minimize the number of search nodes. Min Haitao proposed an enhanced A* algorithm [10], which circumvents collisions by delineating redundant safety spaces and introduces path curvature cost into heuristic function design to improve path smoothness. Cao P et al. introduced an Any-Angle A* algorithm grounded in visibility graphs [11], enabling path searches from any angle, thereby reducing path length and search space. Cheng Chuanqi et al. [12] proposed a strategy for key point selection, aiming to eliminate redundant nodes and unnecessary turning points. In dynamic environments, Likachev et al. introduced the real-time A* search algorithm, known as the D* algorithm [13-14], effectively addressing dynamic alterations. Toll and Geraerts further contributed to this domain by devising a dynamic pruning A* algorithm tailored for replanning in navigation grids [15]. Dakulovi et al. introduced a bidirectional D* algorithm utilizing weighted cost maps for path planning and replanning [16]. However, the D* algorithm and reprogramming methods exhibit limitations in large dynamic environments, with path search times escalating with increasing map size [17].

The Dynamic Window Approach (DWA) algorithm integrates sensor information for real-time local path planning of robots and exhibits excellent dynamic obstacle avoidance capabilities. Missura M proposed an enhanced version of the

Manuscript received on April 15, 2023; revised on December19, 2023. This work was supported in part by the Youth Fund Program of the National Natural Science Foundation of China (Grant No. 61703067), the Chongqing Basic Science and Frontier Technology Research Program (Grant No. Cstc2017jcyjAX0212), and the Science and Technology Research Program of Chongqing Municipal Education Commission (KJ1704072).

Zhangfang Hu is a professor at the Key Laboratory of Optical Information Sensing and Technology, School of Optoelectronic Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065 China; (e-mail: huzf@cqupt.edu.cn)

Xingyuan Wang is a graduate student of the School of Optoelectronic Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065 China; (corresponding author phone: 178-039-06016; e-mail: 17803906016@163.com)

Junhao Zhang is a graduate student of the School of Optoelectronic Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065 China; (e-mail: 769971483@qq.com)

DWA algorithm [18], which incorporates the movement of other objects in the environment. It predicts future collisions by establishing a dynamic collision model, leading to a notable decrease in collision occurrences compared to conventional DWA algorithms. Additionally, Mai Xiquan et al. [19] introduced an evaluation subfunction related to obstacle distribution density in the DWA evaluation function. This enhancement enables the robot to proactively accelerate and avoid entering densely populated obstacle areas. EDUARDO et al. [20] proposed the DW4DO and the DW4DOT methodologies to effectively address dynamic obstacles within the framework of conventional dynamic window algorithms. This modification contributes to an enhancement in the algorithm's safety and stability.

Characterized by an incapacity to discern dynamic obstacles within the environment, the global path planning algorithm is deficient in its ability to dynamically avoid them. Conversely, the local path planning algorithm is vulnerable to converging towards local optima, and its capacity to ensure the realization of a viable solution remains uncertain. Consequently, path planning algorithms tailored for dynamic environments must possess the capability to perform global path planning and dynamic obstacle avoidance simultaneously. In this paper, the improved A* algorithm is initially employed for global path planning. Subsequently, the optimized dynamic window algorithm is utilized to undertake local path planning, taking into account both the global path information and dynamic changes in the environment. This approach aims to enhance the practicability and flexibility of the algorithm.

II. Improved A* Algorithm

The A* algorithm is a graph search algorithm based on heuristic search principles, representing an enhancement of Dijkstra's algorithm. Its foundational principle entails designating the initial parent node as the starting reference point. The algorithm progresses through the exploration of the child nodes in the vicinity of the parent node, systematically calculating the expense value for each child node. The subsequent selection of the next parent node is contingent upon identifying the node with the smallest cost value among the children. This process iterates until the search node encompasses the target point. The ultimate formation of the global path entails connecting from the starting point, the target point, and all parent nodes. The search for child nodes is primarily conducted in four or eight directions, and node management is facilitated through two lists: Openlist and Closelist. Openlist stores the extended child nodes, while Closelist houses obstacle nodes and previously expanded parent nodes. The cost function associated with the node is

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

n is the present node, $g(n)$ denotes the actual cost incurred from the origin to node n, $h(n)$ is the heuristic estimated cost from node n to the final destination. Evaluation function $f(n)$ is used to determine the direction of the search, each time the next node is selected, the subsequent node in the search process is determined by selecting the one with the minimum value of the variable $f(n)$.

The global path produced by the conventional A* algorithm often exhibits numerous redundant nodes, fragmented paths, and substantial turning angles at key nodes, which may not conform to the kinematic principles of mobile robots. In response to these challenges, this paper introduces an enhanced A* algorithm. Initially, a heuristic function is formulated, integrating both Manhattan distance and Euclidean distance, to more accurately approximate the actual cost, thereby enhancing search efficiency. Subsequently, the triangular pruning method is applied to systematically eliminate redundant points in the path, resulting in a reduction in its overall length. Finally, to mitigate the adverse effects of excessive turning angles on the operational stability of mobile robots, the Minimum Snap technique is employed to smooth and optimize the path.

A. Heuristic Function Optimization

The selection of the heuristic function $h(n)$ significantly influences the efficiency and correctness of the A* algorithm. For the assurance of discovering the optimal path, $h(n)$ should not exceed the actual expense from node n to the goal node. $h(n) \le d(n, goal)$, $d(n, goal)$ represents the actual expense of node n to the goal node. The smaller the value of $h(n)$, the more nodes the algorithm will traverse, which in turn will cause the algorithm to run slower. The algorithm's performance improves as the value of $h(n)$ approaches the actual cost.

$$h_m(n) = |x_n - x_{goal}| + |y_n - y_{goal}| \tag{2}$$

$$h_e(n) = ((x_n - x_{goal})^2 + (y_n - y_{goal})^2) \tag{3}$$

x_n and y_n is the abscissa and ordinate of the present node, x_{goal} and y_{goal} are the abscissa and ordinate of the target node, $h_m(n)$ is a heuristic function using the Manhattan distance, $h_e(n)$ is a heuristic function using Euclidean distance. In this study, a heuristic function is formulated by integrating both Manhattan distance and Euclidean distance to closely approximate the actual cost. As the current node moves significantly away from the target node, the Euclidean distance becomes noticeably smaller compared to the actual distance. This discrepancy results in an increased number of search nodes within the algorithm, consequently impeding operational speed. At this time, the weight of $h_m(n)$ should be reduced and the weight of $h_e(n)$ should be increased; when the present node is closer to the goal node, the Euclidean distance is close to the actual distance. At this time, the weight of $h_e(n)$ should be reduced and the weight of $h_m(n)$ should be increased. The improved cost function is shown below.

$$f(n) = g(n) + w(\sqrt{2}p(n)h_e(n) + (1 - p(n))h_m(n)) \tag{4}$$

$$p(n) = \frac{|x_n - x_{start}| + |y_n - y_{start}|}{|x_{goal} - x_{start}| + |y_{goal} - y_{start}|} \tag{5}$$

$p(n)$denotes the fraction representing the proportion of the Manhattan distance from the present node to the goal node in

relation to the total distance, x_{start} and y_{start} are the abscissa and ordinate of the target node, w is the weight of the heuristic function.

In order to substantiate the effectiveness of the enhanced heuristic function, a rigorous experimental evaluation was performed on a 50x50 grid map. With the parameter value configured at 2, the optimized A* algorithm meticulously traversed 451 nodes, whereas its traditional counterpart, employing Euclidean distance, systematically explored 1197 nodes. Notably, in contrast to the extensive node exploration undertaken by the traditional A* algorithm, the optimized variant showcased a marked reduction of 62.3% in the total number of nodes surveyed. The graphical representation of these findings is visually depicted in Figure 1(a) and (b).

B. Redundant Point Elimination Method

The trajectory produced by the A* algorithm typically encompasses redundant nodes, implying that certain nodes along the trajectory can be bypassed without altering the ultimate path outcome. Failure to eliminate these redundant nodes may lead to an increase in the length and complexity of the trajectory, thereby diminishing the path's reliability. The schematic representation of the redundant point elimination process is delineated in Figure 2.

1) Using the path nodes produced by the A* algorithm as input, create and initialize the node set AstarNset. Create a key node set KeyNset to store optimized path nodes. The initial value is $\{n_1, n_m\}$, where n_1 signifies the path's inception, n_m represents the designated target, and m is the number of nodes in the path generated by A*.

2) Connect $n_3, n_4, \cdots, n_i$ sequentially from n_1 to assess whether the straight line $n_1 n_i$ intersects with obstacles. If it passes through obstacles, then n_{i-1} is a key node, and it is added to the KeyNset set; Otherwise, it is judged that $n_2, \cdots, n_i$ is a redundant node, and the nodes in AsatrNset are connected backwards from n_i until it is connected to the target node. At this time, the KeyNset collection contains all key nodes, and the value is $\{n_1, \cdots, n_k, \cdots, n_m\}$.

3) Connect all the key nodes in KeyNset in sequence, so far the optimization is completed.

To validate the effectiveness of the redundant node removal method, a comparative analysis was conducted between the conventional A* algorithm and the A* algorithm enhanced by the redundant point removal method and path pruning. The evaluation was performed on grid maps of dimensions 20 x 20 (Fig 2(a)) and 50 x 50 (Fig 2(b)). The optimized A* algorithm exhibited a noteworthy reduction in turning points, with decreases of 37.5% and 26% for the 20 x 20 and 50 x 50 grid maps, respectively. Additionally, the optimized A* algorithm demonstrated a decrease in path length of 19.3% and 15.1% for the corresponding grid configurations when compared to the traditional A* algorithm.

C. Minimum Snap Smoothing Optimization

Despite the optimization achieved through the redundant point elimination strategy, the resulting path still lacks the desired smoothness. During the execution of this path by the mobile robot, it may encounter numerous sharp turns and require frequent acceleration and deceleration operations. These factors contribute to heightened energy consumption, increased difficulty in robot control, and diminished driving efficiency. In response to this, the Minimum Snap algorithm, a trajectory smoothing technique based on polynomial optimization, is introduced. This algorithm generates a seamlessly smooth trajectory by imposing penalty terms on the derivatives of polynomials. Typically, the trajectory can be effectively represented by an nth-order polynomial.

$$p(t) = p_0 + p_1 t + p_2 t^2 + \ldots + p_n t^n = \sum_{i=0}^{n} p_i t^i \tag{6}$$

$p_0, p_1, p_2, \ldots, p_n$ is a trajectory parameter. Given the inherent simplicity of a polynomial curve, accurately capturing the complexities of a trajectory presents challenges. To address this, a segmentation approach based on time is employed. Each segment is depicted by a polynomial curve to enhance the description of the trajectory.

$$p(t) = \begin{cases} [1, t, t^2, ..., t^n] \cdot p_1 & t_0 \le t < t_1 \\ [1, t, t^2, ..., t^n] \cdot p_2 & t_1 \le t < t_2 \\ \ldots \\ [1, t, t^2, ..., t^n] \cdot p_k & t_{k-1} \le t < t_k \end{cases} \tag{7}$$

k represents the quantity of trajectory segments, and $p_i = [p_{i0}, p_{i1}, p_{i2}, \ldots, p_{in}]$ is the parameter vector of the i-th segment trajectory. Each trajectory is represented by an nth order polynomial with the same time interval. Construct the Minimum Snap optimization function.

$$\min \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} (p^{(4)}(t))^2 dt = \min \sum_{i-1}^{k} p^T Q_i p \tag{8}$$

$$Q_i = \begin{pmatrix} 0_{4\times 4} & 0_{4\times(n-3)} \\ 0_{(n-3)\times 4} & \frac{r!}{(r-4)!}\frac{c!}{(c-4)!}\frac{1}{(r-4)+(c-4)+1}(t^{(r+c-7)} - t_{i-1}^{(r+c-7)}) \end{pmatrix} \tag{9}$$

The matrix is indexed by the row and column values represented by r and c respectively. Solving the entire trajectory to minimize the Snap problem can be converted to solving each curve, $min p^T Q p$ can see that this is a quadratic programming problem. Construct equality constraints for position, first derivative, and second derivative.

$$[1, t_0, t_0^2, \ldots, t_0^n, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = p_0 \tag{10}$$

$$[0, 1, 2t_0, \ldots, nt_0^{n-1}, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = v_0 \tag{11}$$

$$[0, 0, 2, \ldots, n(n-1)t_0^{n-2}, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = a_0 \tag{12}$$

Construct equality constraints for the positions, first derivatives, and second derivatives between adjacent segments. The equality constraints between segment i and segment i-1 are shown below.

$$[\underbrace{0 \ldots 0}_{(i-1)(n+1)}, 1, t_i, t_i^2, ..., t_i^n, -1, -t_i, -t_i^2, ..., -t_i^n, \underbrace{0 \ldots 0}_{(k-1)(n+1)}]p = 0 \tag{3}$$

To assess the efficacy of Minimum Snap smoothing optimization, simulation verification is carried out on two different 10 X 10 grid maps, and the results are shown in Figure 4(a) and (b).

III. Improved DWA Algorithm

The fundamental principle underpinning the dynamic window method revolves around the meticulous definition of the spatial realm wherein a robot can feasibly traverse during the trajectory planning process. This space comprises all points that the robot can reach within a specified time frame. Subsequently, a dynamic window is identified within this feasible space, expressing the spectrum of feasible linear and angular velocities accessible to the robot within the designated time interval, while incorporating safety buffers to account for distances in potential collisions. Utilizing the dynamic window, the motion model is employed for path planning, ultimately resulting in the selection of an optimal trajectory.In this study, we enhance the evaluative mechanism of the established Dynamic Window Approach (DWA) algorithm by integrating supplementary sub-functions, thereby enriching its computational capabilities for trajectory planning. These include an evaluation sub-function related to the distance of dynamic obstacles and another sub-function that specifically characterizes and quantifies the extent of deviation from the predefined global path, thereby contributing to a more refined and comprehensive evaluation within the algorithmic framework. Addressing the single-motion state limitation of the traditional DWA algorithm, we introduce two new motion states, brake waiting and follow-up, to better adapt to dynamic environmental conditions.

A. Robot Motion Model

Forecasting the motion state of the robot is imperative in the dynamic window algorithm, followed by the generation of the motion trajectory according to the sampling speed (v,ω) and the motion model. Under the assumption that the robot is constrained to move solely in the forward direction or undergo rotation, the robot's motion within an extremely brief time interval is approximated as uniform linear motion, and the motion model is shown in formula (14).

$$\begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ \theta(t+\Delta t) \end{pmatrix} = \begin{pmatrix} x(t)+v(t)*\cos(\theta(t))*\Delta t \\ y(t)+v(t)*\sin(\theta(t))*\Delta t \\ \theta(t)+\omega(t)*\Delta t \end{pmatrix} \tag{14}$$

B. Velocity Sampling

In accordance with the limitations imposed by the robot's performance characteristics and environmental obstacles, the velocity space is constrained to obtain a dynamic window of velocity. The mobility of the robot is constrained by both its maximum and minimum achievable speeds, and the corresponding constraints are shown in formula (15).

$$V_s = \{(v,\omega) \mid v \in [v_{\min}, v_{\max}] \wedge \omega \in [\omega_{\min}, \omega_{\max}]\} \tag{15}$$

V_s is the set of speeds that the robot can achieve. Affected by the performance of the motor, the mobility of the mobile robot is constrained by the upper limits of acceleration and deceleration, and the corresponding constraints are shown in formula (16).

$$V_d = \{(v,\omega) \mid v \in [v_c - \dot{v}_b \Delta t, v_c + \dot{v}_a \Delta t] \wedge \omega \in [\omega_c - \dot{\omega}_b \Delta t, \omega_c + \dot{\omega}_a \Delta t]\} \tag{16}$$

The velocity parameter V_d is governed by limitations imposed by both the upper bounds of acceleration and deceleration, $\dot{v}_b$ denotes the upper limit for linear velocity acceleration, whereas $\dot{\omega}_b$ indicates the maximum acceleration achievable for angular velocity in this context. To prevent collisions with obstacles, the speed is subject to constraints as shown in equation (17).

$$V_a = \left\{(v,\omega) \mid v \le \sqrt{2dist(v,\omega)\dot{v}_b} \wedge \omega \le \sqrt{2dist(v,\omega)\dot{\omega}_b}\right\} \tag{17}$$

The velocity space $\{(v,\omega) \mid (v,\omega) \in V_s \cap V_d \cap V_a\}$ satisfying the three constraints constitutes the dynamic window of velocity.

C. Improved Evaluation Function

The speed is sampled in the dynamic window, estimating the motion trajectory involves applying the motion model, introducing an evaluation function, scoring the trajectory, and ultimately selecting the optimal trajectory. Because the traditional dynamic window method is prone to converging towards local optima, the integration of global path information is performed in this paper to establish the sub-function for evaluating the global path. The improved evaluation function is shown in formula (18).

$$\begin{aligned} G(v,\omega) = \sigma(&\alpha * heading(v,\omega) + \beta * dist(v,\omega) \\ &+ \mu dist_m(v,\omega) + \gamma vel(v,\omega) \\ &+ \psi Path(v,\omega)) \end{aligned} \tag{18}$$

$heading(v,\omega)$ serves as the azimuth evaluation function, assessing the angular disparity between the final orientation of the currently predicted trajectory and the target point. $dist(v,\omega)$ and $dist_m(v,\omega)$ are obstacle distance evaluation functions, $dist(v,\omega)$ measures the spatial separation between the terminal point of the currently projected trajectory and stationary obstacles, and $dist_m(v,\omega)$ assesses the proximity of the current predicted trajectory's end to dynamic obstacles. $vel(v,\omega)$ as the speed evaluation metric, scrutinizing the robot's current linear speed to facilitate an expeditious approach towards the target point. $Path(v,\omega)$ as the evaluation metric for global path offset, determining the distance between the endpoint of the current predicted trajectory and the global path.

D. Optimized motion state

In the conventional dynamic window method, the motion process usually revolves around a singular state, wherein the robot navigates towards the target point while consistently evading obstacles. The robot maintains its trajectory until the

attainment of the target point. To overcome this limitation, this paper introduces the notion of behavior states, integrating two supplementary motion states: braking waiting and maintaining following. Through the utilization of information derived from behavior states, the optimized dynamic window method can adeptly strategize obstacle avoidance paths, thereby diminishing turning angles and augmenting the overall performance of obstacle avoidance.

In situations involving a dynamic L-shaped trap, the conventional dynamic window method may give rise to suboptimal trajectories characterized by pronounced turning angles or close alignment with the obstacle's direction of motion. To mitigate this challenge, the present study proposes the incorporation of a motion state denoted as "brake waiting." This state is activated contingent upon the satisfaction of the following conditions: (1) The moving obstacle is positioned in the forward trajectory of the robot; (2) The proximity of the robot to the obstacle is less than a specified threshold, denoted as d1; (3) The speed of the obstacle surpasses a defined threshold, denoted as v1; (4) The angle deviation between the obstacle's motion direction and the robot's planned trajectory surpasses 45°, as depicted in Figure 5. Under these prescribed conditions, the robot undergoes a temporary cessation of movement, affording the moving obstacle the opportunity to vacate the planned path before recommencing its trajectory. The parameters d1 and v1 can be subject to customization to align with the specific operational exigencies of the robotic system.

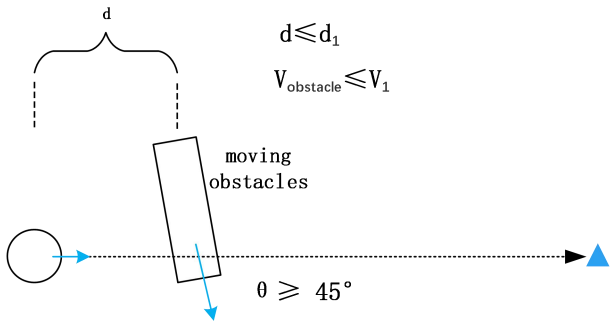


Figure.5. the motion state of brake waiting

When a moving obstacle is positioned ahead of the robot, and its trajectory closely aligns with or mirrors that of the robot, the traditional dynamic window method typically resorts to circumventing the obstacle to the left or right. However, in practical movement scenarios, it is often more optimal for the robot to follow the path of the obstacle. To address this, our paper introduces an additional follow-up motion state under the following conditions: (1) The moving obstacle is situated along the global path ahead of the robot; (2) The proximity to the robot is within a distance less than d2; (3) The velocity of the obstacle exceeds 2/3 of the robot's maximum speed constraint; (4) The angular disparity between the obstacle's trajectory and the present orientation planned by the robot is less than 25°, as depicted in Figure 6. During such circumstances, the robot maintains its original route and sets the maximum speed constraint to match the moving velocity of the obstacle. The value of parameter d2 is subject to adjustment based on the unique circumstances of the robot.

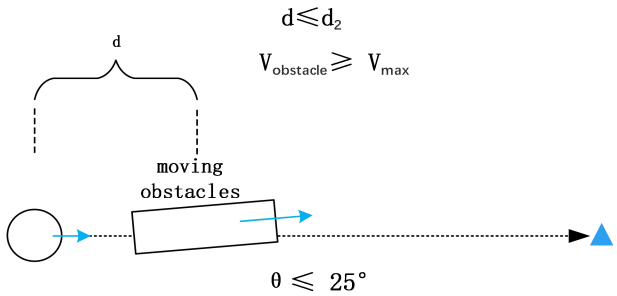


Figure.6. the follow-up motion state

To confirm the improved performance of the refined DWA algorithm in dynamic obstacle avoidance, an experimental simulation is carried out. Let ψ be 0, temporarily disregarding the deviation of the planning path from the global trajectory. The red square denotes a moving obstacle, and the red dotted line is its trajectory. In Figure 7(a), the angle formed by the robot and the dynamic obstacle exceeds 45° , prompting the robot to enter a braking waiting state, anticipating the obstacle's clearance from the path. Within the transition from Figure 7(a) to Figure 7(b), the angular difference between the robot and the dynamic obstacle is below 25° , and the robot maintains a slightly adjusted original route. Ultimately reaching the designated target point, as depicted in Figure 7(d).

IV. Fusion Algorithm

Within a dynamic environmental context, it is imperative for robots to promptly perceive alterations within their surroundings and make corresponding decisions in real-time. The A* algorithm utilizes a guiding function to assess the expense associated with the proximity of each node to the target. Subsequently, it selects the subsequent node by considering both the cost and the path taken to arrive there. While the A* algorithm guarantees the discovery of the shortest path, its practical application for real-time navigation may be constrained by map complexity and computational resources.

Conversely, the Dynamic Window Approach (DWA) algorithm empowers robots to dynamically adjust their speed and angular velocity within the dynamic environment in response to perceived dynamic obstacles. If the perception system detects an obstacle approaching the robot's path, the DWA algorithm can opt to modify the robot's speed or angular velocity to avert potential collisions.

The fusion of an enhanced A* algorithm with the Dynamic Window Algorithm allows the robot to simultaneously consider the global optimal path and engage in dynamic obstacle avoidance throughout the path planning process. Initially, The global optimal path is determined through the utilization of the improved A* algorithm. Subsequently, select key nodes within the global path node sequence as local target points guiding the Dynamic Window Approach in the formulation of the local path. The workflow of the integrated algorithm is illustrated in Figure 8.

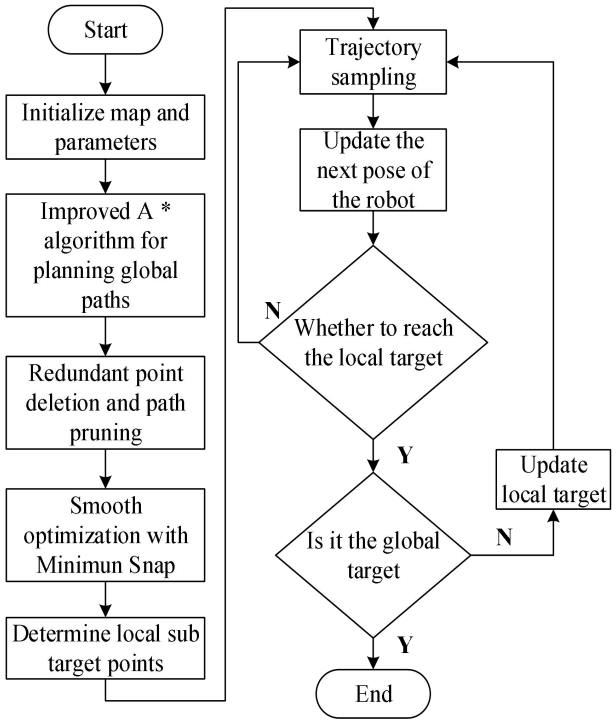


Figure.8. Fusion algorithm flowchart

V. Simulation and Analysis

To validate the efficacy of the enhanced A* algorithm integrated with the Dynamic Window Algorithm (DWA), the fusion algorithm is simulated and verified within the Matlab 2022b environment. Two sets of simulation experiments are configured—one under static environmental conditions and the other under dynamic environmental conditions.

A. Simulation Analysis of Fusion Algorithm in Static Environment

The path planning process is conducted on 20x20 and 30x30 grid maps, utilizing the traditional A* algorithm, the enhanced A* algorithm, the conventional DWA algorithm, and the novel fusion algorithm introduced in this study. On the 20x20 grid map, coordinates $(1m,1m)$ mark the initial position, and coordinates $(20m,20m)$ represent the destination, with an obstacle coverage rate of 17.1%. On the 30x30 grid map, coordinates $(30m,30m)$ represent the designated target location, with an obstacle coverage rate of 17.6%. The motion parameters for the robot in the conventional Dynamic Window Algorithm are: the highest linear velocity is $2m/s$, the highest angular velocity is $0.35rad/s$, the acceleration is $0.2m/s^2$, and the rotation acceleration is $0.8rad/s^2$. The motion parameters in the fusion algorithm are consistent with those of the conventional DWA algorithm. The outcomes of path planning are illustrated in Figure 9, accompanied by a performance comparison of the four algorithms presented in Table 1.

After analyzing the data presented in Table 1,Demonstrating a noteworthy reduction in path length by 12.02% and 8%, respectively, the enhanced A* algorithm stands out in comparison to the conventional A* algorithm. And resulting in a substantial reduction in search nodes by 61% and 58.88%, respectively. However, The induction of collision scenarios is observed in both the upgraded A* algorithm and the conventional A* algorithm, posing a potential risk to the safety of the robotic system.

On the contrary, while the traditional DWA algorithm prioritizes path safety and dependability, it demonstrates path lengths on the two grid maps that are 1.46 times and 1.68 times greater than those achieved by the enhanced A* algorithm. In this paper, we propose a fusion algorithm that utilizes the enhanced A* algorithm for initial path planning, followed by the seamless integration of the improved DWA algorithm to manage obstacle avoidance in the immediate vicinity. Consequently, it preserves the advantageous characteristics of the improved A* algorithm regarding search node count and path length, all the while facilitating efficient maneuvering around local obstacles.

B. Simulation Analysis of Fusion Algorithm in Dynamic Environment

To gauge the effectiveness of the fusion algorithm in random obstacle avoidance, two distinct series of simulation experiments were executed on a 30x30 grid map. One set pertained to a solitary undisclosed obstacle, whereas the other encompassed multiple unidentified obstacles. The simulation results are illustrated in Figure 10, and comprehensive simulation data can be found in Table 2. Notably, the yellow squares in the figures represent random obstacles introduced subsequent to the preliminary phase of path planning in the enhanced A* algorithm.

In Figure 10(a), a solitary random obstacle is positioned along the path pre-determined by the enhanced A* algorithm. Employing the global path node as the local target point for path planning, the robot temporarily diverges from the globally optimal trajectory due to the imperative of avoiding random obstacles. In Figure 10(b), two random obstacles are introduced based on the configuration in Figure 10(a), with one of them positioned at a certain distance from the globally optimal trajectory and not impacting the path planning of the fusion algorithm. Another random obstacle is in proximity to the optimal global trajectory, and the fusion algorithm needs to temporarily stay away from the global path to avoid this obstacle. According to Table 2, in order to avoid the third random obstacle within the fusion algorithm, the length of the path is 0.83m longer, and the deviation from the path arc is 0.2rad. In conclusion, the presented fusion algorithm in this paper effectively achieves avoidance of random obstacles, and the planned path closely approximates the globally optimal trajectory.

To assess the fusion algorithm's capability in avoiding dynamic obstacles, dynamic elements are introduced onto the 30x30 grid map. The improved DWA algorithm, the potental field ant colony method [21] and the proposed fusion algorithm in this study are simulated and subjected to comparison. The red square is a dynamic obstacle, and the red dotted line is its

trajectory. The outcomes of the simulations are illustrated in Figure 11, with corresponding data detailed in Table 3.

Figure 11 illustrates the efficacy of the enhanced DWA algorithm, the Potential Field Ant Colony (PFAC) algorithm, and the algorithm presented in this investigation for dynamic obstacle avoidance. The PFAC algorithm's obstacle-avoidance path is minimally impacted by dynamic obstacles; however, it exhibits non-smooth trajectories with numerous abrupt changes in direction. The improved DWA algorithm, on the other hand, skillfully navigates around dense obstacle regions, but it results in the longest path length. The data presented in Table 3 clearly indicates that the fusion algorithm introduced in this study not only achieves a path with smooth obstacle avoidance but also boasts the shortest path length among the assessed algorithms. In contrast to the improved DWA algorithm and the PFAC algorithm, it results in a reduction of path length by 25.8% and 12.97%, respectively.

VI. Conclusion

This paper introduces a innovative method for dynamic obstacle avoidance in robotics, which integrates an enhanced A* algorithm with an optimized Dynamic Window Approach (DWA). This integration effectively addresses several challenges prevalent in this domain, including issues related to search efficiency, path smoothness, superfluous nodes, and adaptability to navigating arbitrary obstacles within complex environments.The algorithm manifests an autonomous adjustment mechanism for the weighting of the heuristic function contingent upon the robot's spatial relationship with the goal, thereby substantiating the heightened efficacy of the search process. It introduces a technique for eliminating redundant path points, resulting in reduced path length, and incorporates Minimum Snap optimization to enhance path trajectory smoothness. Moreover, The A* algorithm is improved to discern crucial nodes within the generated path node sequence, serving as local destination points within the optimized DWA algorithm. Endowing the robot with the capability to navigate both random and dynamic obstacles, this feature enhances its adaptability in diverse environments.

In static environments, the conventional A* and DWA algorithms, along with their improved counterparts and the fusion algorithm introduced in this investigation, are subjected to simulation. The results suggest that the fusion algorithm not only preserves the benefits of the improved A* algorithm concerning search node count and path length but also surpasses in local obstacle avoidance. Furthermore, the algorithm demonstrates improved path security and real-time performance.

In dynamic settings, extensive simulations and meticulous testing were carried out to systematically assess the effectiveness of the improved DWA algorithm, the PFAC algorithm, and the fusion algorithm introduced in this study. The results underscore the outstanding efficacy of the fusion algorithm in he realm of dynamic obstacle evasion.

References

[1] T. T. Mac, C. Copot, D. T. Tran, et al., "A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization," *Applied Soft Computing*, vol. 59, pp. 68-76, 2017.

[2] B. Y. Song, Z. D. Wang, L. Zou, "On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm," *Cognitive Computation*, vol. 9, no. 1, pp. 5-17, 2017.

[3] M. A. Rendón, F. F. Martins, "Path following control tuning for an autonomous unmanned quadrotor using particle swarm optimization," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 325-330, 2017.

[4] B. K. Patle, D. R. K. Parhi, A. Jagadeesh, et al., "Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot," *Computers & Electrical Engineering*, vol. 67, pp. 708-728, 2018.

[5] B. Fu, L. Chen, Y. Zhou, et al., "An improved A* algorithm for the industrial robot path planning with high success rate and short length," *Robotics and Autonomous Systems*, vol. 106, pp. 26-37, 2018.

[6] S. M. Persson, I. Sharf, "Sampling-based A* algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683-1708, 2014.

[7] H. M. Zhang, M. L. Li, L. Yang, "Safe path planning of mobile robot based on improved A* algorithm in complex terrains," *Algorithms*, vol. 11, no. 4, p. 44, 2018.

[8] D. Šišlák, P. Volf, M. Pechoucek, "Accelerated A* trajectory planning: Grid-based path planning comparison," in *Proceedings of the 19th International Conference on Automated Planning & Scheduling (ICAPS)*, Menlo Park, CA, USA: AAAI Press, 2009, pp. 74-81.

[9] M. Lin, K. Yuan, C. Shi, et al., "Path planning of mobile robot based on improved A* algorithm," in *Proceedings of the 2017 29th Chinese Control and Decision Conference*, Piscataway, NJ: IEEE, 2017, pp. 3570-3576.

[10] H. Min, X. Xiong, P. Wang, et al., "Autonomous driving path planning algorithm based on improved A* algorithm in unstructured environment," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 2-3, pp. 513-526, 2021.

[11] P. Cao, Z. Fan, R. X. Gao, et al., "A focal any-angle path-finding algorithm based on A* on visibility graphs," *arXiv preprint arXiv:1706.03144*, 2017.

[12] Chuan-qi Cheng, Xiang-yang Hao, Jian-sheng Li, et al., "Global dynamic path planning based on fusion of improved A* algorithm and dynamic window method," *Journal of Xi'an Jiaotong University*, vol. 51, no. 11, pp. 137-143, 2017. (in Chinese)

[13] M. Likhachev, S. Koenig, "A Generalized Framework for Lifelong Planning A* Search," in *ICAPS*, 2005, pp. 99-108.

[14] M. Likhachev, D. Ferguson, G. Gordon, et al., "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613-1643, 2008.

[15] W. van Toll, R. Geraerts, "Dynamically Pruned A* for re-planning in navigation meshes," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 2051-2057.

[16] M. Dakulović, I. Petrović, "Two-way D∗ algorithm for path planning and replanning," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 329-342, 2011.

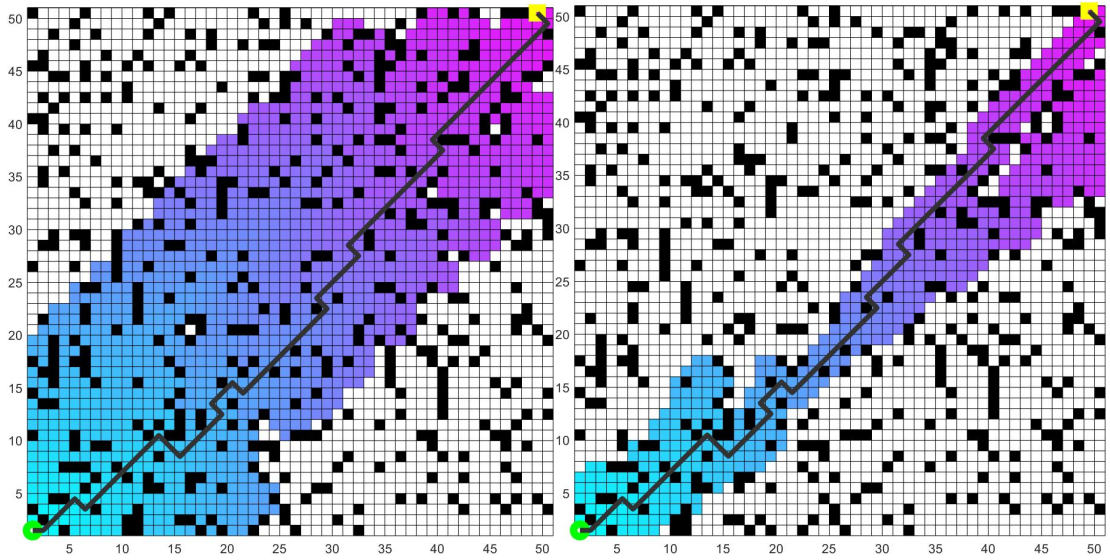
[17] A. Ammar, H. Bennaceur, I. Châari, et al., "Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments," *Soft Computing*, vol. 20, pp. 4149-4171, 2016.

[18] M. Missura, M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8620-8626.

[19] X. Mai, D. Li, J. Ouyang, et al., "An improved dynamic window approach for local trajectory planning in the environment with dense objects," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, vol. 1884, no. 1, p. 012003.

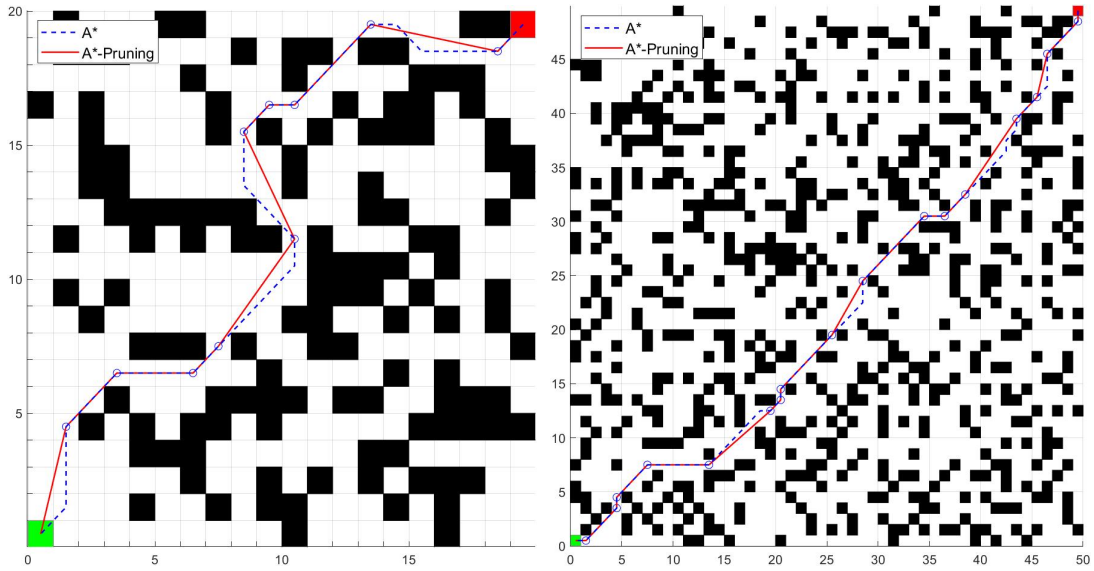
[20] E. J. Molinos, A. Llamazares, M. Ocaña, "Dynamic window based approaches for avoiding obstacles in moving," *Robotics and Autonomous Systems*, vol. 118, pp. 112-130, 2019.

[21] L. Jianhua, Y. Jianguo, L. Huaping, et al., "Robot global path planning based on ant colony optimization with artificial potential field," *Nongye Jixie Xuebao/Transactions of the Chinese Society of Agricultural Machinery*, vol. 46, no. 9, 2015.



(a) Traditional A* Algorithm (b) Optimize A* Algorithm

Figure.1. Heuristic function optimization



(a) 20X20 Grid Map (b)50X50 Grid Map

Figure.3. Redundant point elimination and path pruning

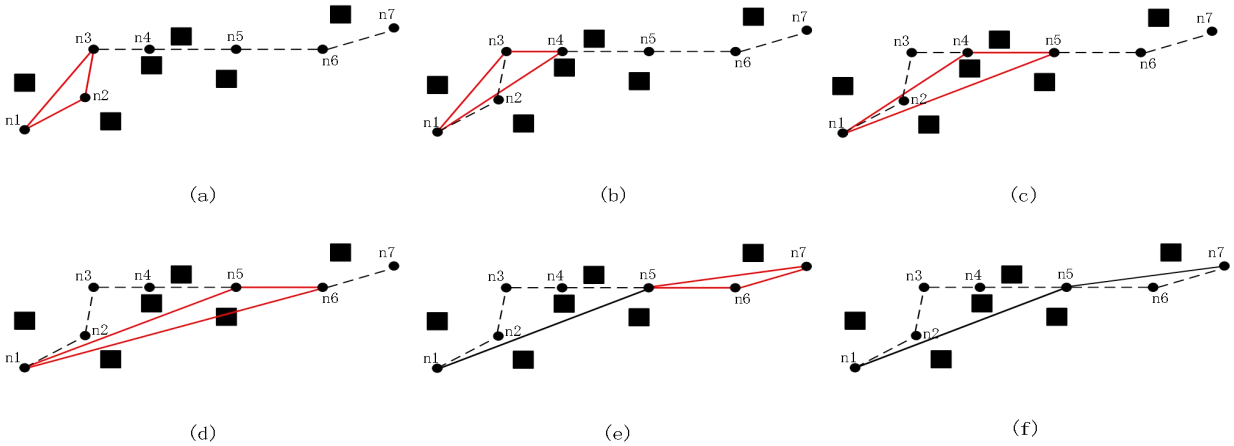
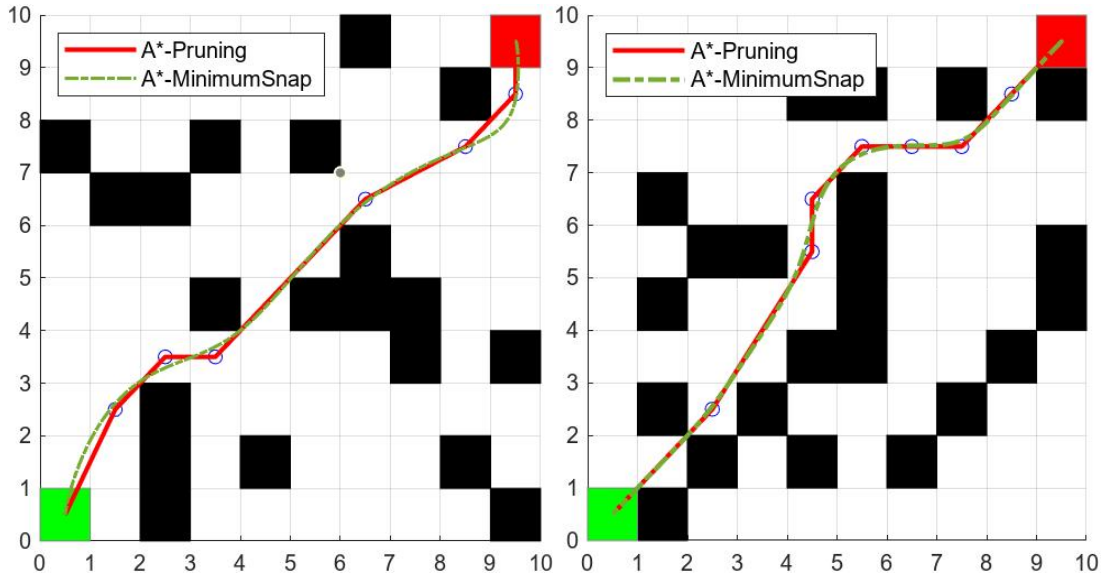
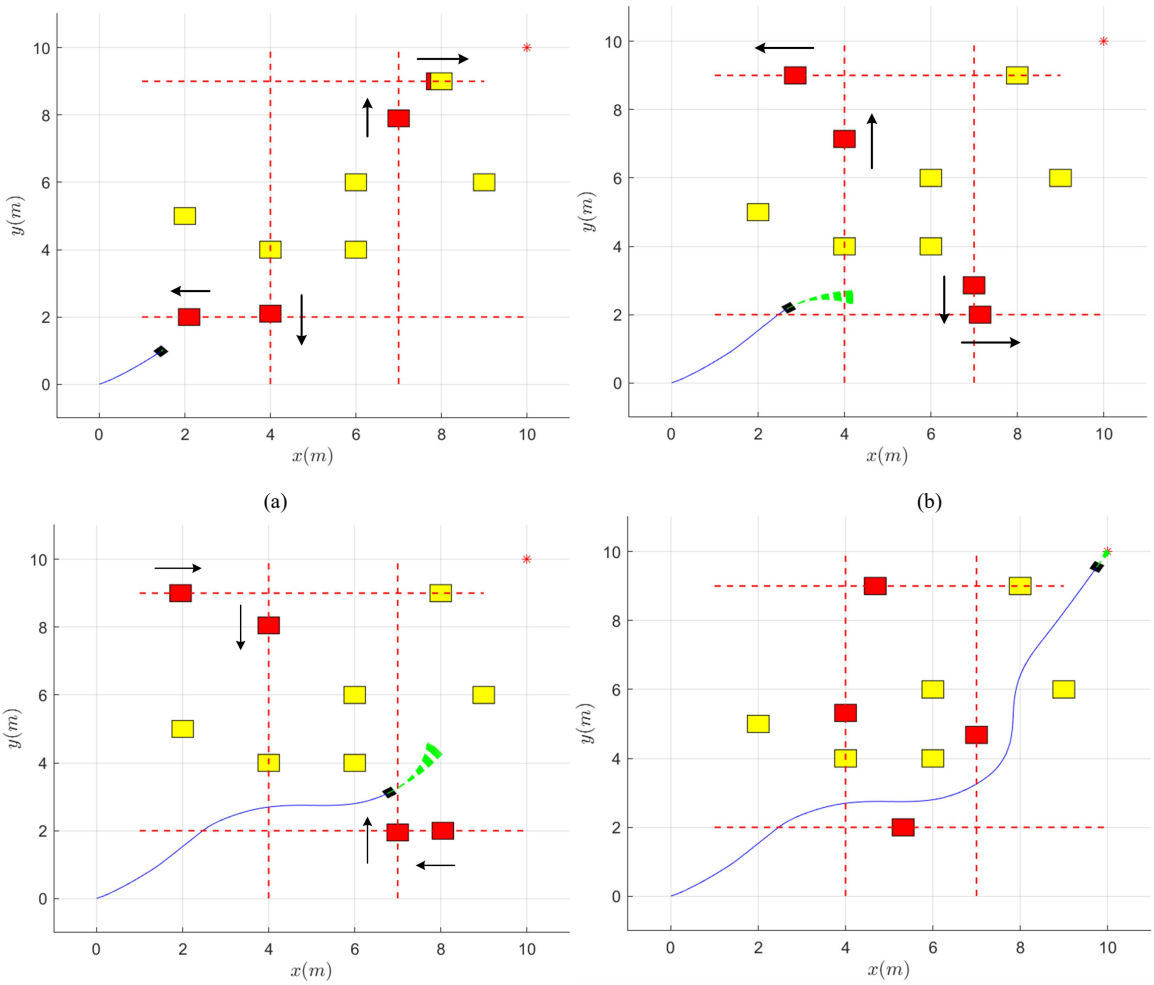


Figure.2. Redundant point elimination process



(a)10X10 Grid Map 1 (b)10X10 Grid Map 2

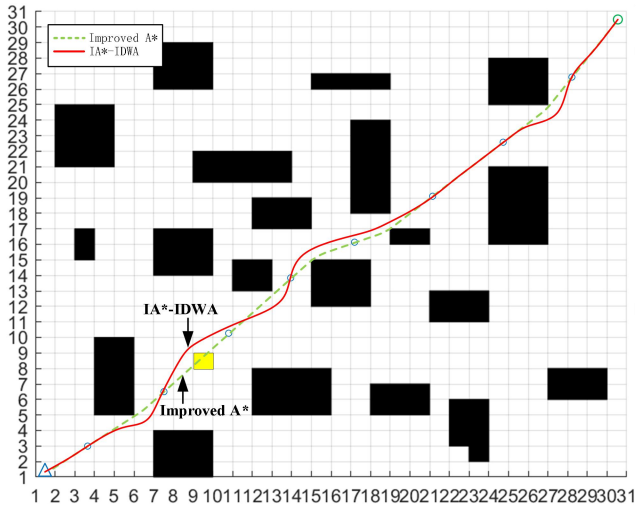
Figure.4. Minimum Snap smoothing optimization



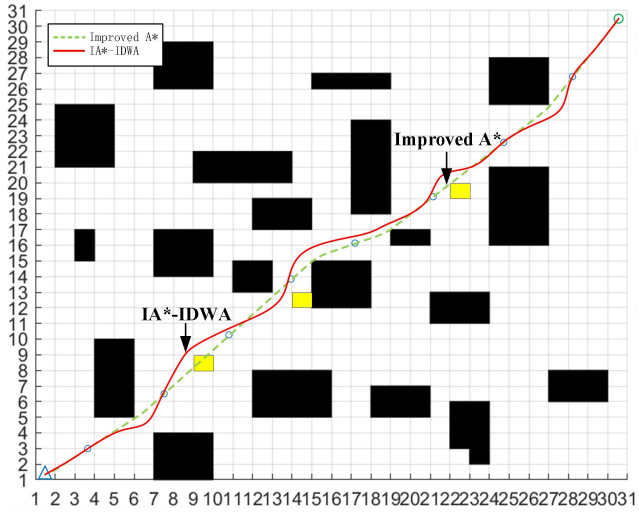
(a) (b)

(c) (d)

Figure.7. Improved DWA algorithm dynamic obstacle avoidance



(a) an obstacle

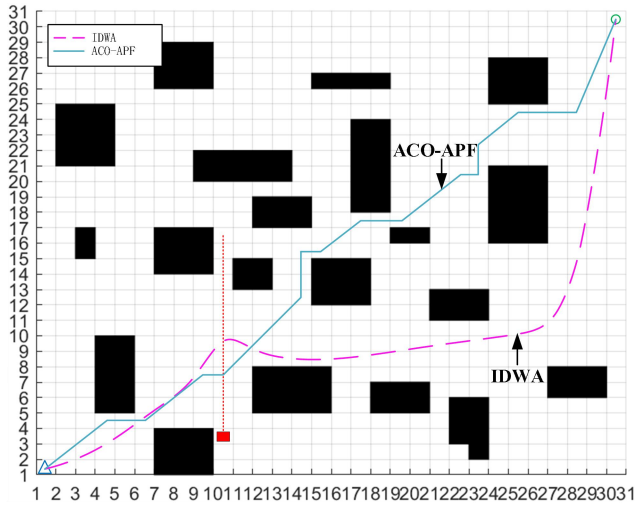


(b) three obstacles

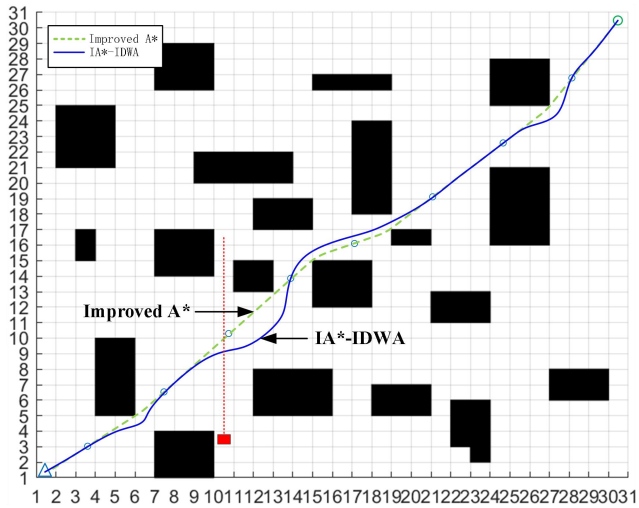
Figure.10. Random obstacle avoidance simulation

TABLE II
RANDOM OBSTACLE AVOIDANCE SIMULATION DATA

Number of unknown obstacles	Whether to successfully avoid obstacles	Length(m)	Cumulative turning arc (rad)
1	√	43.95	1.6
3	√	44.78	1.8



(a) ACO-APF and Improved DWA

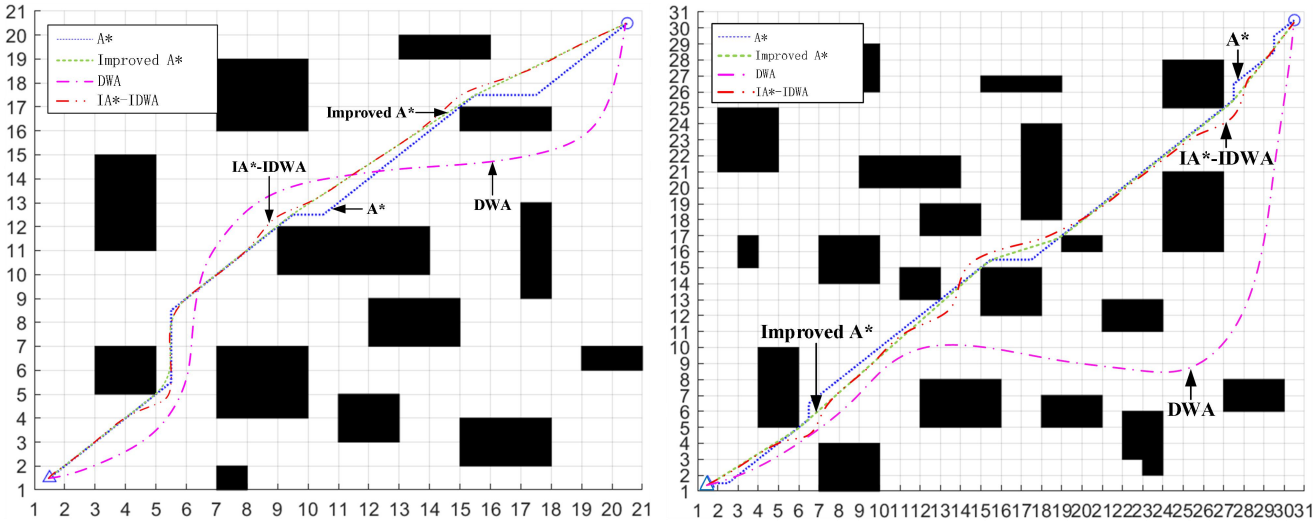


(b) Fusion Algorithm

Figure.11. Dynamic obstacle avoidance simulation

TABLE III
SIMULATION DATA OF DYNAMIC OBSTACLE AVOIDANCE

Algorithm	Length(m)	Whether to successfully avoid obstacles	Smooth Path
Improved DWA	59.16	√	√
ACO-APF	50.42	√	x
Fusion Algorithm	43.88	√	√



(a)20X20 Grid Map 1 (b)30X30 Grid Map 2

Figure.9. Simulation results in static environment

TABLE I
SIMULATION DATA IN STATIC ENVIRONMENT

Map	Algorithm	Expanded Nodes	Smooth Path	Safety	Length
20x20	Traditional A*	192	x	x	28.63
	Improved A*	75	√	x	25.19
	Traditional DWA		√	√	36.69
	Fusion Algorithm	75	√	√	27.06
30x30	Traditional A*	428	x	x	42.77
	Improved A*	176	√	x	39.35
	Traditional DWA		√	√	65.63
	Fusion Algorithm	176	√	√	43.21