IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 10, 2021, accepted January 23, 2021, date of publication January 28, 2021, date of current version February 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3055231

A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A-Star for Mobile Robots

ZHIHAI LIU[1], HANBIN LIU[1], ZHENGUO LU[1], AND QINGLIANG ZENG[2,3]

[1]College of Transportation, Shandong University of Science and Technology, Qingdao 266590, China
[2]College of Mechanical and Electronic Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[3]College of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

Corresponding author: Qingliang Zeng (qlzeng@sdust.edu.cn)

This work was supported in part by the Key Research and Development Plan of Shandong Province under Grant 2019GGX104048 and Grant 2019SDZY01, in part by the Ministry of Education of China under Grant IRT_16R45, and in part by the National Natural Science Foundation of China under Grant 51674155.

ABSTRACT Although many studies exist on mobile robot path planning, the disadvantages of complex algorithms and many path nodes in logistics warehouses and manufacturing workshops are obvious, mainly due to the inconsistency of map environment construction and pathfinding strategies. In this study, to improve the efficiency of mobile robot path planning, the Delaunay triangulation algorithm was used to process complex obstacles and generate Voronoi points as pathfinding priority nodes. The concept of the grid was used to extract obstacle edges to provide obstacle avoidance strategies for robot pathfinding. Subsequently, the search for priority and regular path nodes used the improved A-star (A*) algorithm. The dynamic fusion pathfinding algorithm (DFPA), based on Delaunay triangulation and improved A*, was designed, which realizes the path planning of mobile robots. MATLAB 2016a was used as the simulation software, to firstly verify the correctness of the DFPA, and then to compare the algorithm with other methods. The results show that under the experimental environment with the same start point, goal point, and number of obstacles, the map construction method and pathfinding strategy proposed in this paper reduce the planned path length of the mobile robot, the number of path nodes, and the cost of overall turn consumption, and increase the success rate of obtaining a path. The new dynamic map construction method and pathfinding strategy have important reference significance for processing chaotic maps, promoting intelligent navigation, and site selection planning.

INDEX TERMS Delaunay triangulation, A-star algorithm, mobile robot, map modeling, path planning.

I. INTRODUCTION

The development of robotics and in-depth research of automatic navigation technology have promoted the widespread application of mobile robots in various fields. Map construction and path planning are the core and research focus of mobile robot development and design. In a robot working environment with static obstacles, mobile robots can avoid obstacles and move from the start position to the target position. The shortest path length and the minimum energy consumption is the research goal. At present, most studies are based on the detection of obstacles for multiple path planning [1]–[3], or local path correction in an unknown working environment to avoid obstacles [4]–[8]. The overall process is complicated, and dynamic integration of map construction and path planning is rarely achieved. The dynamic combination of map modeling and path planning can accomplish the collaborative goal of avoiding obstacles using path planning, simplify algorithm complexity [6], reduce the detection of path nodes, and improve the efficiency of path planning.

Mobile robots equipped with various types of sensors can realize path planning, motion control, and data transmission. The mobile robot converts the entire working environment into a two-dimensional map based on sensor positioning. According to the principle of geometry, the operating system regards the mobile robot as a point and realizes the path planning of the mobile robot from the start point to the target point [9]. The most direct method is visibility-based

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

graphs, connect the vertices of obstacles to construct different polygons [10]–[12], and then use path planning algorithms to plan the path. The Dijkstra algorithm is a typical method and realizes the shortest path search by accumulating the path length. The Rapidly-exploring Random Tree algorithm (RRT) adopts the concept of branches, extending from the start point and continuously expanding the random tree without encountering obstacles until it reaches the target point. The RRT is suitable for complex obstacle environments, but because the optimal solution cannot be quickly converged [10], and the calculation time is long [11]. The Dijkstra and RRT approaches use obstacle information for direct path planning. Artificial potential field algorithm (APF) no longer uses obstacle information directly but introduces the potential field concept to process obstacle information. The APF was proposed to establish an attractive potential field at the target point and a repulsive potential field around the obstacle. Path planning was achieved through the control of the total potential field [13]. The APF needs to define many parameters of the potential field to ensure the performance of the algorithm. Changes in the obstacle environment will cause the parameters not always to be optimal, so the adaptability of APF is insufficient [12]. Thus, improvements of the APF that incorporate methods such as optimal trade-off functions have been developed [13], [14]. Dijkstra, RRT, and APF are path planning algorithms based on the environment. If no further map processing takes place, problems arise such as a large number of nodes, a long calculation iteration time, and poor adaptability.

The common methods for processing maps are the grid method and the Voronoi diagram. The grid method can grid the working environment, and is a convenient and straightforward approach [2], [15]. The Voronoi diagram generated by Delaunay triangulation [16]–[18] can order the complex obstacle environment, and is widely used in the environment modeling of mobile robots [19]–[21]. After the map is processed, the common path planning algorithm includes the D* algorithm, Genetic Algorithm (GA), Ant Colony Optimization Algorithm (ACO), and A*. The D* algorithm can realize path planning by judging different path points in the map grid. D* has good adaptability and is often used for rapid replanning [2]. The GA was first proposed in 1975, and is a group search algorithm based on the principle of chromosome evolution [22]. It is widely used in path planning, but has the disadvantages of slow convergence and local inability to reach optimality. The Bezier curve was introduced into the GA to improve the path trajectory [6]. The main method to improve the optimization and adaptability of the GA is to improve the compilation operator and coding method [23]. The ACO is inspired by the ant foraging behavior in nature, looking for the best path from the nest to the food. Ants communicate with each other by placing pheromones in the footprint. The ACO will set the parameters based on the pheromone. The root cause of local stagnation lies in the limitations of the algorithm search mechanism. The A* algorithm was first proposed in 1968 [24]. As a heuristic algorithm, it can implement a global search. Related research has resulted in the A* algorithm being widely used in different environments [4], [8], [15], [19], [20], [25]–[27]. The combination of the A* algorithm and the grid method is the most common method, which is likely to cause too many turns, and the path is too tortuous, so improving the smoothness of the path is the main research content. Song *et al.* [27] studied the path planning of an unmanned boat in an outdoor environment and improved the path node optimization of the A* algorithm, making the entire path smoother and the reducing the overall energy consumption. Phan Gia Luan et al. used a multi-level A* algorithm, adding the Bézier curve in the first layer to achieve path smoothing, and using a weighted model of dynamic obstacles in the second layer to achieve obstacle avoidance [28]. Improving the path's smoothness will significantly reduce the number of turns and the robot's turning angle, thereby saving energy consumption. Since the A* algorithm is a global search algorithm, A* will detect all path nodes, which ensures the algorithm's reliability, but detecting too many path nodes also results in slow operation. Kala *et al.* [8] combined the A* algorithm with the fuzzy inference system to improve the accuracy of path selection. The fuzzy cost function was used to analyze obstacle avoidance data to achieve optimal path selection. Yin and Yang *et al.* [25] added a delete algorithm to the A* algorithm path selection to improve the extent of the search of the A* algorithm, reducing the negative impact of large-scale networks and achieving control of traffic. Fu *et al.* [26] simplified the A* algorithm and added a pre-processing stage to achieve a path from the starting point to the end point; using direct judgment in the post-processing stage, the path length was reduced by optimizing the path node. Reducing the useless path node detection in the A* pathfinding process can effectively improve A* flexibility. Combining the grid method and A* algorithm is challenging to reduce path node detection, but the Voronoi diagram has a good effect. The Voronoi diagram is a more complicated map processing method than the grid method. Many scholars have done much research on the application of the Voronoi diagram in mobile robot path planning. Reducing the useless path node detection in the A* pathfinding process can effectively improve A* flexibility. Combining the grid method and A* algorithm is challenging to reduce path node detection, but the Voronoi diagram has a good effect. The Voronoi diagram is a more complicated map processing method than the grid method. Many scholars have done much research on the application of the Voronoi diagram in mobile robot path planning. In 2017, Candeloro *et al.* [20] changed the map modeling method of the A* algorithm, using the Voronoi diagram as the map of A* path node selection, and realized the theoretical verification of the A* algorithm in the Voronoi diagram. In 2019, based on the Voronoi map modeling, Ayawli *et al.* [19] used the A* algorithm for initial path planning, and then classified the obstacles according to the dynamics and dynamic obstacles of the mobile robot, and realized the obstacle avoidance path optimization of the mobile robot. The improvement of

the map modeling method promotes the improvement of A*. The A* is often used in combination with other path planning algorithms, and the fusion application of multi-path planning algorithms is a research trend. In 2015, Zuo *et al.* [15] adopted the hierarchical A*, using the A* algorithm for the initial path planning at the first layer, and using the approximate strategy for local path correction at the second layer to achieve obstacle avoidance and smoothing of the path. Many studies have proved the combination of the A* algorithm, and the grid method is a common method [29]. The main research content is to improve the path's smoothness and reduce the number of path nodes [30]. The map has the feature of ordering. The combination of the A* algorithm and Voronoi diagram greatly improves the path's quality, but the map modeling method cannot recognize the shape of the obstacle, and the obstacle avoidance strategy is very troublesome [31].

This study aimed to design a new map modeling and improve the A* algorithm to improve the autonomous operation efficiency of mobile robots and provide a reference for collaborative control of mobile robot systems. In this paper, we propose the dynamic fusion pathfinding algorithm (DFPA) based on Delaunay triangulation and an improved A* algorithm. The newly designed map modeling method combines the Voronoi diagram and the grid method. The Delaunay triangulation algorithm is used to deal with obstacle coordinates and generates an ordered Voronoi diagram. The Voronoi point is generated as a preferred alternative path node. At the same time, the obstacles are placed in the grid according to the shape and location, and the edges of the obstacles are extracted. To improve the A* algorithm, we introduced direct judgment into A*, and designed a new pathfinding process to reduce the number of iterations. Furthermore, we also designed a new obstacle avoidance method using the obstacle side of the map modeling environment, improved the obstacle avoidance and path optimization. Finally, we introduced the Steiner point to optimize the path in reducing the path turning consumption and length. This paper is a path planning method after the Voronoi diagram and grid method. Compared with the visibility-based graphs [12], [13], it has more flexibility in environmental recognition. The difference between the method proposed in this paper and the common Voronoi diagram combined with the A* algorithm [19], [20] is that we add a grid to extract all obstacle edges based on the Voronoi diagram, which can identify the shape of obstacles. Compared with other improved A* algorithms [25], [27], [28], we draw on the idea of path pre-judgment [26] and summarize the pre-judgment and the Voronoi point as the priority pathfinding nodes into the pathfinding process. In addition, compared to local path correction [15] and classified obstacle avoidance [19], we use the obstacle edges of map modeling as the basis for obstacle avoidance. The new obstacle avoidance strategy uses map modeling data.

The contributions and novelties of the paper are as follows:

1) One of this paper's contributions is to design a map modeling method that combines the Voronoi diagram and grid method. Using the Voronoi diagram alone for map modeling can reduce path node detection, but it is difficult to identify the shape of obstacles, while the grid method alone can take into account the area of obstacles but will cause too many path nodes. The newly designed map modeling method fully combines the advantages of the two methods. It introduces the ordered Voronoi points into the path planning, avoids excessive path node detection, and can use the grid to identify the area of obstacles and extract obstacle edges.

2) Other contribution is the new pathfinding process and obstacle avoidance strategy to improve the A* algorithm. Introducing the directness judgment in advance into the A* algorithm can avoid many calculations to find the direct path from the start point to the target point. In addition, we combine the three methods of direct judgment, finding the Voronoi point, and traditional A* finding the node into a new pathfinding process, which reduces the complexity of iteration. The obstacle avoidance strategy relies on the obstacle edges generated by map modeling, which is simple.

3) The innovation of this paper is to design a new map modeling method that can be integrated with the improved A* algorithm, which can avoid repeated calculations, maximize the use of data resources generated in the map modeling process, and improve the pathfinding process and simplify iteration complexity.

The remainder of the paper is organized as follows. Section II introduces the concepts of the A* and Delaunay triangulation algorithms, the characteristics of the Voronoi diagram, and the basic principles of map modeling and recognition of obstacles. Section III introduces the flow and content of the entire DFPA. Section IV simulates and verifies the DFPA, and compares the experimental results of the DFPA with A* and RRT under the same conditions. Section V discusses the DFPA in depth. Finally, Section VI provides the conclusions of the study.

II. DYNAMIC FUSION PATHFINDING ALGORITHM METHOD

A. DELAUNAY TRIANGULATION AND A* ALGORITHM

The generation of triangular meshes is a key part of many computational modeling and simulation tasks, such as computational engineering, numerical simulation, and computer graphics. Triangular meshes are essentially a combination of different complex points. The Delaunay triangulation algorithm is often used to regularize complex and irregular points. Delaunay triangulation generates triangle meshes without overlapping by connecting sampling points scattered in the problem domain [16]. This method has been proved to be effective in processing complex points. The Voronoi diagram is generated on the basis of Delaunay triangulation, which is often used for sensor placement in the field of geology and surveying [17]. Delaunay triangulation theory has been thoroughly studied [18], [32], [33].

The A* algorithm is a heuristic algorithm that searches for the path with the lowest cost among all possible paths [8], that searches for nodes in the graph from the start node to

the target node, and that uses heuristics related to information about the problem features to guide its performance. As a heuristic algorithm, the A* algorithm includes three parts: OPEN LIST, CLOSED LIST, and heuristic function, as shown in Equation (1):

$$F = G + H \tag{1}$$

where F is the overall consumption cost, G is the actual consumption cost, and H is the heuristic cost. (1) is the core of the A* algorithm. F is used to evaluate the current node and offer a search order among candidate nodes for the next search [26]. The heuristic function is the basis of the A* algorithm search path node. Often, the actual consumption cost cannot be changed, and is difficult to improve. Generally, the improvement of the A* algorithm is studied from the heuristic cost H [15], and H is optimized and improved according to different usage environments to improve the A* algorithm. Since we take the Voronoi point as a priority pathfinding node, and the mobile robot moves on the Voronoi diagram, the path length is the main cost. We define the path length from the start point to the current node as the actual consumption, which can measure the current node's actual performance. We use the Euclidean distance from the current node to the target point to define, which can reflect the current node's priority in the next search. In this way, the node with the smallest will be selected next time.

The premise of mobile robot path planning is the construction of maps, that is, the analysis of the working environment based on environmental perception. The most important issue in the path planning process is real-time positioning, including the positioning of mobile robots and obstacles. Delaunay triangulation can be used to process complex data and obstacle coordinate points, so that complex and disordered obstacle coordinate points can be standardized for computer simulation. As the main method of static map construction, the number of grids will increase the number of calculations to be performed by the mobile robot. Accurate location and navigation is a challenge, and result in the robots' moving path being overly complicated and involving too many turns. A map construction method based on Delaunay triangulation to deal with obstacles can overcome the shortcomings of the grid method. The A* algorithm has good pathfinding efficiency. Thus, combining the approach of the A* algorithm's pathfinding with Delaunay triangulation for environment map construction, we designed a new pathfinding strategy. In this study, obstacle grid filling was used in map modeling, and A* was used in the pathfinding strategy to expand and find path nodes. Furthermore, the obstacle coordinates were converted into grid positions, and the grid positions were converted into coordinates. This data conversion method makes it possible to dynamically combine the two elements of map modeling and pathfinding.

B. THE OPTIMAL POSITION OF THE VORONOI EDGES

Through the Delaunay triangulation algorithm, a set of points can be generated into a unique map. Because this unique map is composed of triangles, it is also called Delaunay triangular grid map. The Delaunay triangle grid map satisfies the following rules:

1) Any edge does not contain a point in the point set;
2) All edges have no intersecting edges.

The Delaunay triangle grid map is shown in Figure 1. The black solid points are obstacles and the black dotted lines form the Delaunay triangle grid. This kind of triangle mesh map has a significant advantage: it integrates complex and disordered point sets into regular edges. The prerequisite for a mobile robot to realize path planning is to recognize complex obstacles. The Delaunay triangular grid map can be used as a method of map modeling for mobile robots.

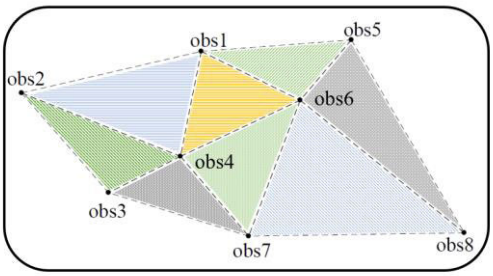

FIGURE 1. Delaunay triangle grid map.

Although the Delaunay triangle grid map transforms the complex points into regular edges, which is not suitable for robot path movement, the Voronoi diagram on this basis does have good characteristics. The Voronoi diagram includes Voronoi points and Voronoi edges. The Voronoi point is the center point of all Delaunay triangles that meet the requirements [34]:

1) Any two Delaunay triangles have a common side;
2) The center of the circumcircle of the two Delaunay triangles conforming to 1) is the Voronoi point;
3) The centers of the circumcircles of the two Delaunay triangles that conform to 1) are connected to form a Voronoi edge.

The Voronoi diagram is shown in Figure 2.

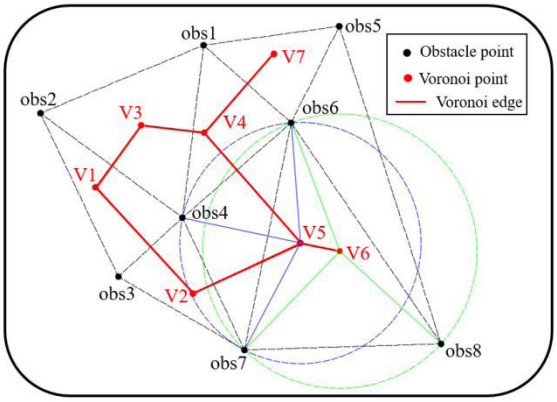

FIGURE 2. The voronoi diagram.

The typical feature of the Voronoi diagram is that the sum of the distances from all points on the Voronoi edge to all vertices of two adjacent Delaunay triangles is the smallest, as shown in Figure 3.

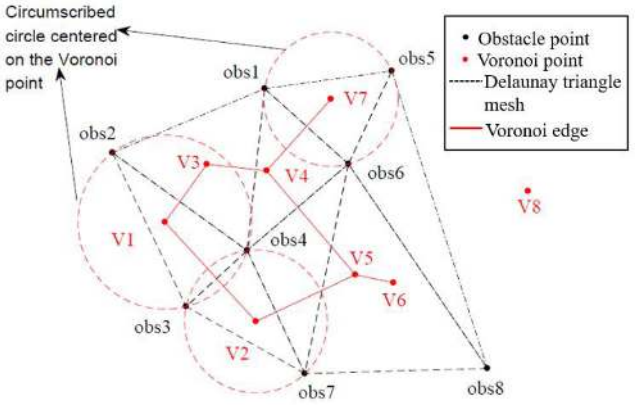

FIGURE 3. The distance from the point on the Voronoi edge to all vertices of two adjacent Delaunay triangles is the smallest.

All obstacle coordinate points can be expressed as Equation (2).

$$obs = \{(s\_x_1, s\_y_1), (s\_x_2, s\_y_2), \cdots\cdots, (s\_x_{m-1}, s\_y_{m-1}), (s\_x_m, s\_y_m)\} \quad (2)$$

The Voronoi points can be expressed as Equation (3).

$$VP = \{(v\_x_1, v\_y_1), (v\_x_2, v\_y_2), \cdots\cdots, (v\_x_{n-1}, v\_y_{m-1}), (v\_x_n, v\_y_n), \} \quad (3)$$

As shown in Figure 4, the distance from $V1$ to three obstacle points $\{obs_3, obs_2, obs_4\}$ is $\{dis_{1-3}, dis_{1-2}, dis_{1-4}\}$, respectively, and the distance from $V2$ to three obstacle points $\{obs_4, obs_7, obs_3\}$ is $\{dis_{2-4}, dis_{2-7}, dis_{2-3}\}$, respectively. Any point $\{point_1, point_2, \cdots, point_i, \cdots\cdots\}$ on the Voronoi edge is composed of $V1$and V2, and the distance between $point_i$ and the obstacle on both sides of the Voronoi edge is equal, which means that the Voronoi edge is always in the middle of the obstacle points. In Equation (4),dis_{i-3} is the distance from $point_i$ on the Voronoi edge to obs_3, and dis_{i-4} is the distance from $point_i$ on the Voronoi edge to obs_4

$$dis_{i-3} = dis_{i-4} \quad (4)$$

Not every obstacle point can generate a Delaunay triangle and not every circumscribed circle center of each Delaunay triangle (Voronoi point) can be connected to form a Voronoi edge. According to the rules, n(n $\leq$ m) Delaunay triangles can be generated, and n(n $\leq$ m) Voronoi points can be generated simultaneously. We know that the Voronoi edges and the obstacle points form a special topological relationship graph, and the Voronoi edges divide the entire area into multiple block structures. The Voronoi edge always occupies the center of obstacles on both sides. The closed-loop Voronoi edge contains only one obstacle point. The movement of the mobile robot on the Voronoi edges ensures that it is always

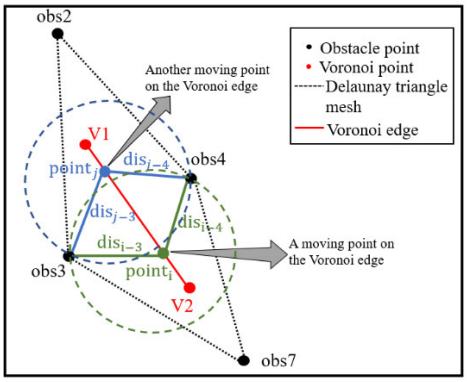

FIGURE 4. Any point on the Voronoi edge is always in the middle.

in the middle of the obstacle. The Voronoi diagram integrates disordered obstacle points into an ordered topological relationship diagram, as shown in Figure 5. Therefore, we use the Voronoi diagram as a method for mobile robots to process obstacle points.

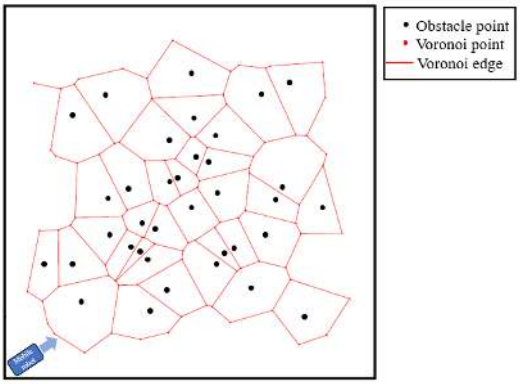

FIGURE 5. A special topological relationship graph.

C. OBSTACLE COORDINATE VISUALIZATION

The Voronoi diagram can handle complex and disorderly obstacles, and the Voronoi edge has the excellent characteristic of always being in the middle of the obstacles on both sides, so we use the Delaunay triangulation algorithm to process obstacle coordinate points and generate a special topological relationship graph that contains obstacle points, Voronoi points, and Voronoi edges. However, the obstacle is an irregular entity with an area instead of a point. In the traditional mobile robot planning algorithm, the grid method can clearly express the grid occupied by the obstacle. The number and position of the grid represent the occupation of the obstacle. Therefore, on the basis of generating a Voronoi diagram according to the coordinates of obstacle points, we introduce the concept of a grid to realize the performance of the occupied area of obstacles, improve the construction of the map, and make early preparations for obstacle avoidance.

The grid method has the characteristics of simple construction and easy and programming and can reflect the area of

obstacles. On the basis of the concept of the traditional grid method, we combine the principle of the optimal position of the Voronoi edge, the size of the mobile robot, and the actual distribution of obstacles to achieve obstacle grid filling.

Suppose the boundary of the entire working area is:

$$\begin{cases} X_{min}; & X_{max} \\ Y_{min}; & Y_{max} \end{cases} \tag{5}$$

and we randomly generate N points as obstacle coordinates:

$$obs = rand(N, 2) \tag{6}$$

According to the principle of Voronoi graph generation, we can generate Voronoi points and Voronoi edges.

$$VP = \begin{Bmatrix} v\_x_1 & v\_y_1 \\ \vdots & \vdots \\ v\_x_n & v\_y_n \end{Bmatrix} \tag{7}$$

$$Voronoiedgemat = \begin{Bmatrix} VP_1(x) & VP_1(y) & VP_2(x) & VP_2(y) \\ VP_3(x) & VP_3(y) & VP_4(x) & VP_4(y) \\ \vdots & \vdots & \vdots & \vdots \\ VP_{\mu-1}(x) & VP_{\mu-1}(y) & VP_{\mu}(x) & VP_{\mu}(y) \end{Bmatrix} \tag{8}$$

N obstacles generate n Voronoi points and μ Voronoi edges.

The size of the mobile robot is $L * W$.

The traditional grid method is divided according to the length and width of the mobile robots. We then select the largest size of the mobile robot as the basic unit division of the grid, and the number of grids is:

$$Numgrid = \{\max(X_{max} - X_{min}, Y_{max} - Y_{min})/[\max(L, W)]\}^2 \tag{9}$$

The Voronoi points have precise coordinates, which are displayed in the grid graph and used as the background data. Placing obstacles in a grid is the key element of using Voronoi diagrams for map modeling, and also an important strategy for mobile robots to avoid obstacles.

Although the shapes of obstacles are different (irregular), the grid is square (regular). In the process of converting obstacle coordinates into the grid map, the area of the obstacle is converted into the number and approximate location of the grid. We use the geometric center of the obstacle as its coordinate point, as shown in Figure 6.

According to the shape of the obstacle and the grid occupation situation, all grids that contain obstacle edges and points are obstacles, which enlarge the actual area occupied by obstacles. It will not lead to inaccurate path planning, but avoid excessively complicated calculations. After determining the coordinates of the geometric center point of the obstacle and estimating the shape, we can determine the number and position of the grid occupied by the obstacle, and further clarify the obstacle edges. Fig.6 is taken as an

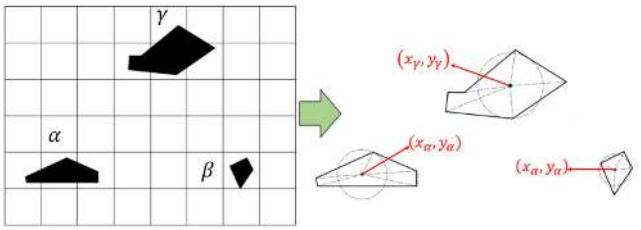

FIGURE 6. Obstacle coordinates and shape diagram.

example. The shapes of the three obstacles are different, and the geometric center coordinate points are:

$$\begin{aligned} obs_\alpha &= (s\_x_\alpha, s\_y_\alpha) \\ obs_\beta &= (s\_x_\beta, s\_y_\beta) \\ obs_\gamma &= (s\_x_\gamma, s\_y_\gamma) \end{aligned} \tag{10}$$

According to the geometric center point and shape of the obstacle, we can obtain the grid number and position occupied by each obstacle and the edges of each obstacle. The coordinates of the four vertices of the edges of the α obstacle are:

$$\begin{aligned} obsxmin_a &= ceil\,(obs_\alpha) - numl_a \\ obsxmax_a &= ceil\,(obs_\alpha) + numr_a \\ obsymin_a &= ceil\,(obs_\alpha) - numd_a \\ obsymax_a &= ceil\,(obs_\alpha) + numu_a \end{aligned} \tag{11}$$

where *ceil* is a round-up function, $numl_a$ is the number of grids occupying the grid from the geometric center point of the α obstacle to the left, $numr_a$ is the number of grids occupying the grid from the geometric center point of the α obstacle to the right,$numd_a$ is the number of grids occupying the grid from the geometric center point of the α obstacle to the down, and $numu_a$ is the number of grids occupying the grid from the geometric center point of the α obstacle to the up.

The edges for the α obstacle are:

$$obsedge_\alpha = \begin{Bmatrix} obsxmin_a, & obsymin_a, & obsxmin_a, & obsymax_a \\ obsxmin_a, & obsymax_a, & obsxmax_a, & obsymax_a \\ obsxmax_a, & obsymax_a, & obsxmax_a, & obsymin_a \\ obsxmax_a, & obsymin_a, & obsxmin_a, & obsymin_a \end{Bmatrix} \tag{12}$$

According to (11) and (12), the obstacle edges of all obstacles can be obtained in (13), as shown at the bottom of the next page.

We map the geometric center point of the obstacle to the grid map, and according to the actual shape of the obstacle, we can obtain the position and number of the grid occupied by the obstacle. Through the number and position of the occupied grids and the coordinates of the geometric center point, we further regularize the obstacles and obtain the obstacle edges, which are used as an important reference standard for

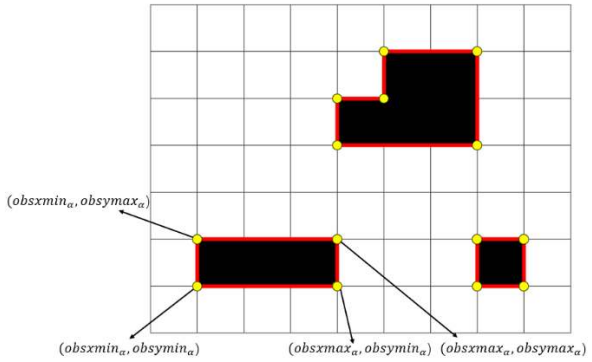

FIGURE 7. α **obstacle edges.**

obstacle avoidance. We intersect the Voronoi edges and the Voronoi points with the obstacle edges to determine the best strategy without hitting the obstacle.

III. DYNAMIC FUSION PATHFINDING ALGORITHM CONTENT

A. DYNAMIC FUSION PATHFINDING ALGORITHM FLOW

The DFPA contains two components: map modeling and improved A* pathfinding. Map modeling and pathfinding are carried out at the same time in the algorithm process because the resources of the improved A* algorithm come from the obstacle edges, Voronoi points, and Voronoi edges of map modeling. The improved A* algorithm takes Voronoi points as priority pathfinding points. In addition, we added direct judgment and designed a pathfinding process to improve the A* algorithm. The pathfinding process includes direct pathfinding, Voronoi point pathfinding, and traditional A* pathfinding. At the same time, the obstacle edges are extracted as a strategy for avoiding obstacles. We take the grid where the obstacle is located as the "wall" of the traditional A* algorithm.

Map modeling provides two data sources for pathfinding, namely, obstacles and Voronoi points. The Voronoi points have both precise coordinate points and row and column values of the grid. Obstacles have precise geometric center coordinate points, grid index values, and obstacle edges.

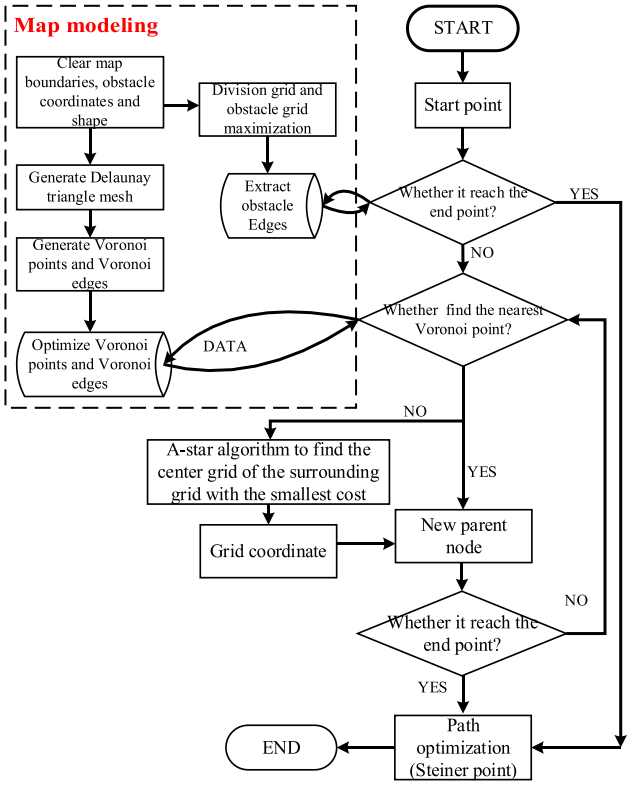

FIGURE 8. **Dynamic fusion pathfinding algorithm flow chart.**

The path search from map modeling is mainly based on the following principles: 1) avoiding obstacles and ensuring that the moving path cannot intersect with the obstacle edges; 2) preferentially finding the Voronoi points as the path nodes, because the Voronoi edge has excellent characteristics, according to (4); and 3) preferring to choose a straight path instead of a polyline path to reduce energy consumption and shorten walking time. Based on the principle of pathfinding and the principle of early map modeling, we designed the DFPA. The DFPA flow chart is shown in Figure 8.

It can be seen from Fig.8 that the map modeling uses the Delaunay triangulation algorithm to generate the Delaunay triangle mesh map and the Voronoi diagram to identify

$$obsedgemat = \begin{Bmatrix} obsxmin_1, & obsymin_1, & obsxmin_1, & obsymax_1 \\ & \vdots & & \\ obsxmin_a, & obsymin_a, & obsxmin_a, & obsymax_a \\ obsxmin_a, & obsymax_a, & obsxmax_a, & obsymax_a \\ obsxmax_a, & obsymax_a, & obsxmax_a, & obsymin_a \\ obsxmax_a, & obsymin_a, & obsxmin_a, & obsymin_a \\ & \vdots & & \\ & \vdots & & \\ obsxmax_N, & obsymax_N, & obsxmax_N, & obsymin_N \\ obsxmax_N, & obsymin_N, & obsxmin_N, & obsymin_N \\ obsxmin_N, & obsymin_N, & obsxmin_N, & obsymax_N \end{Bmatrix} \tag{13}$$

obstacles, but not all Voronoi diagrams can be used. The Voronoi points beyond the boundary are unavailable and will affect the later pathfinding, so map optimization is essential. Simultaneously, the grid will also be divided, and obstacles will be converted into obstacle edges. The Voronoi point and obstacle edges are obtained as essential parameters of the pathfinding strategy. The pathfinding strategy is based on the improved A* algorithm. The main idea is to detect the surrounding path nodes from the start point, find the best path node, and then iterative calculations until the end point is reached. The pathfinding strategy takes the Voronoi point as the priority path node, which provides a way to directly reach the destination, thereby avoiding complicated iterative calculations. The obstacle avoidance strategy determines whether the next path node can be reached. The main idea is whether the path is composed of the current node and the next node intersects the obstacle edges. Later, we will introduce DFPA in detail.

B. MAP OPTIMIZATION

A group of obstacles can generate different Delaunay triangle grid graphs. Some of the Voronoi diagrams generated on this basis may not be used for path planning. In the process of map modeling, we use Delaunay triangulation to generate Voronoi points and Voronoi edges that may exceed the motion boundary, which should be removed in the initial optimization stage.

According to (5), (7), and (8), the optimized Voronoi points can be obtained:

$$VP = (X_{min} < VP_i(1) < X_{max}, Y_{min} < VP_i(2) < Y_{max}) \tag{14}$$

At the same time, we delete the Voronoi edges that are beyond the boundary.

According to the points and shapes of obstacles, we can generate the number of rows and columns of the grid occupied by the obstacles and the index value. On this basis, we can generate the obstacle edges of all obstacles, as in (13). The Voronoi points must be reflected in the grid for later calculation of overall consumption cost. The rasterization of Voronoi points adopts the principle of grid unification, that is, all Voronoi points falling within the same grid are uniformly expressed as the same grid. The conversion of Voronoi points into index value of grids can be expressed as Equation (15):

$$VPindex = [ceil(VP)] \tag{15}$$

The grid occupied by the Voronoi points—the number of rows and columns and the index value—are not important data in the algorithm process, but are used only to calculate statistical movement costs after the algorithm is completed.

C. PATHFINDING STRATEGY

After the initialization phase of the map modeling is completed, we specify the start point and end point using precise coordinates:

$$S = (x_x, y_s)$$
$$E = (x_e, y_e) \tag{16}$$

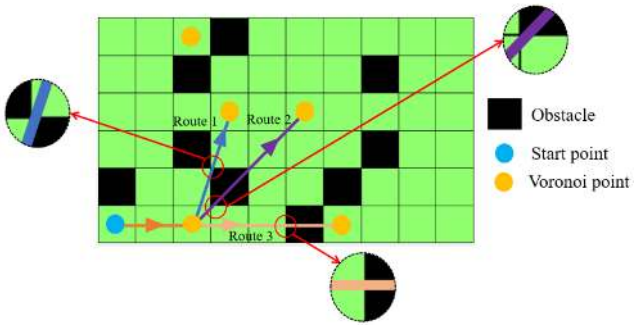

FIGURE 9. The deadlock state.

The traditional A* algorithm detects the surrounding path nodes from the start point until reaching the end point. This algorithm has a large amount of calculation, which will severely limit the performance of the algorithm. When the number of obstacles is small, the A* algorithm still has to perform tedious iterative calculations and is not flexible. In order to improve these shortcomings, we use the Voronoi points generated in the previous map modeling as the priority pathfinding nodes. The start point is no longer looking for the surrounding path nodes, but judging whether it can reach the end point and finding the nearest Voronoi point. Judging whether it can directly reach the end point saves many calculations. The Voronoi point's priority can ensure that the mobile robot moves in the best position, avoiding excessive path node detection. Our main contribution is to provide a correct and effective path planning method.

In the traditional A* algorithm, it is necessary to establish a grid cost database to provide the consumption cost of each grid; the grid cost consumption value occupied by obstacles is infinite. In the pathfinding process of the algorithm, a deadlock phenomenon will occur during the process of finding a new Voronoi point, and the current node can neither directly reach the end point nor find a new Voronoi point, as shown in Figure 9. Therefore, in the case of a deadlock, the cost database of the four grids around the node must be set to use the traditional A* algorithm to find new points to jump out of the deadlock state. The grid cost data is:

$$Field = (Numgrid, Numgrid) + rand(Numgrid, Numgrid) \tag{17}$$

The pathfinding strategy is improved from that used in the traditional A* algorithm, where we establish two sets, OPEN SET and CLOSE SET:

$$Open = \{S, \cdots\cdots\}$$
$$Close = \{\} \tag{18}$$

The OPEN SET is put into the start point and other points to be detected, and the CLOSE SET is put into the parent

node that has been detected as the mobile node of the robot. According to the algorithm flowchart, we make a direct judgment to connect the start point to the end point to detect whether there is a collision with the obstacle edges. If no collision occurs, it is directly connected and the algorithm ends. Otherwise, the seek operation is performed, and all points except the obstacle point and the obstacle grid are used as alternative points. However, based on the particularity of the Voronoi point, the Voronoi point is the preferred point. Therefore, the parent node (start point) searches for the nearest Voronoi point that can be connected and the connecting line does not collide with the obstacle edges.

We determine the Manhattan distance between the current parent node and all Voronoi points, where *abs* is the absolute value function:

$$\begin{aligned} VPH_i &= abs\left(parentnode(1) - VP_i(1)\right) \\ &\quad + abs\left(parentnode(2) - VP_i(2)\right) \qquad (19) \\ NewVP &= \min \quad (VPH_i) \qquad (20) \end{aligned}$$

We also put *NewVP* as the new parent node in the OPEN SET and continue the pathfinding.

If the current parent node cannot find the nearest Voronoi point without touching the obstacle grid, the current parent node is put as the center point. According to the traditional A* algorithm, the grids in the four directions of up, down, left, and right are used to try to find the new path. The grid occupied by obstacles is regarded as a wall and the cost as infinite so that it is not considered. According to the grid generation value of (17) and the Manhattan distance from the try point to the end point, the center point of the grid with the lowest cost can be found as the parent node and placed in the CLOSE SET.

$$costs = min \begin{Bmatrix} Sofarcosts + Field\ (1) + H\ (1) \\ Sofarcosts + Field\ (2) + H\ (2) \\ Sofarcosts + Field\ (3) + H\ (3) \\ Sofarcosts + Field\ (4) + H\ (4) \end{Bmatrix} \qquad (21)$$

We then loop through the entire pathfinding process until the new parent node can be directly connected to the end point without encountering obstacles. All parent nodes in the CLOSE SET are connected in reverse order from the end point to the start point, and the entire path of the robot can be found. If the OPEN SET is completely empty, and the CLOSE SET still has no end point, it means that there is no solution and no feasible path.

D. OBSTACLE AVOIDANCE

In the whole process of the DFPA, the most important issue is to judge whether the parent node can be connected with the new point without touching the obstacle. The obstacle avoidance strategy determines the feasibility of the DFPA.

Avoiding obstacles is one of the main tasks for mobile robot path planning. The typical way is to define a safe distance, and then use sensors to detect the distance between the obstacle and the robot [14]. Another method is to establish a numerical matrix and set the value of the obstacle to infinity, through continuous calculation, to avoid the maximum amount of calculation [8]. The obstacle avoidance strategy proposed in this paper is to record all obstacle edges after the obstacle is identified, and then judge whether the detected path and the obstacle edge intersect, to realize obstacle avoidance. This obstacle avoidance strategy does not rely on sensors, nor does it require many complex calculations using numerical matrices. Obstacle avoidance strategies are more uncomplicated and can be dynamically combined with map modeling.

During the construction of the map, we generated the obstacle edges according to the geometric center coordinate point and shape of the obstacle. Without considering the size of the mobile robot, it can be concluded that the connection between the parent node and the try point does not have any intersection with the obstacle edges, and the path from the parent node to the try point can be realized.

The obstacle avoidance strategy is used to judge whether the line segment formed by the coordinates of the known point P and the try point T intersects the obstacle edges.

$$\begin{aligned} P &= (x_p, y_p) \\ T &= (x_t, y_t) \\ Tryside &= (P, T) \\ Result &= sum\left[cross\left(Tryside, obsedge_i\right)\right] \qquad (22) \end{aligned}$$

where *cross* is a cross-product function. We combine the current parent node and the try point into a new line segment *Tryside*, and the cross-product operation of *Tryside* and each of the obstacle edges. If the result is 0, they will not intersect; and if the result is 1, they will intersect. The sum of the results of each operation are summed:

$$Result = \begin{cases} 0, & Able\ to\ find\ a\ new\ point \\ \rho, & Collision\ with\ obstacle\ \rho\ times \end{cases} \qquad (23)$$

E. PATH OPTIMIZATION

Although the obtained path avoids obstacles, too many turns result in too long path length, which is not the best path. Therefore, we have added path optimization in the last step of DFPA. Bhattacharya, Priyadarshi *et al.* introduced the Steiner point into the Voronoi diagram path to reduce the turn and path length [35] and achieved good results. We optimize the path by the Steiner point.

Steiner points have high stability and are often used for path planning and multi-node optimization. The Steiner point is defined as follows:

$$S = \frac{1}{2\pi} \sum_{j=1}^{f_0} v_j \varphi \left(F_j^0, P\right) \qquad (24)$$

where, P is a set of points, v_j is the coordinates of each vertex F_j^0 $(j = 1, \cdots, f_0)$, and $\varphi\left(F_j^0, P\right)$ is the complementary angle of vertex F_j^0, which satisfies $\sum_{j=1}^{f_0} \varphi\left(F_j^0, P\right) = 2\pi$, then S is the Steiner point of the P.

The final path is connected by many nodes, including the center point of the grid and the Voronoi point. We treat these nodes as a set of points and then calculate Steiner points (24).

Step 1: We set an interval value τ, set many Steiner points on the path's edge based on the interval. Find a path node V, add Steiner points on both sides of the path through V, add the first Steiner point at τ distance away from V, and the second Steiner point at 2τ distance away from V, and so on. We connect the first Steiner points on the two sides. If the connecting line cannot intersects the obstacle edges, we try to connect the second Steiner points. Continue the whole process until the Steiner points reach the turning point. We replace the path node V with the connecting line of the last Steiner points. If the Steiner points on both sides cannot be connected, we retain V.

Step 2: If the shortest path cannot be achieved with the interval value: τ, we shorten the interval value to half: $\frac{\tau}{2}$. We add Steiner points again according to the new interval value, and repeat the above steps to find new connecting lines to replace the current path node. In the whole process, the interval value changes dynamically, interval value $= \frac{\tau}{4}, \frac{\tau}{8}, \frac{\tau}{16} \cdots\cdots$.

Repeat the Step 1 and Step 2 until the last interval value reaches the predetermined minimum value, achieve the maximum resolution, then end the entire optimization process.

Finally, we delete unnecessary Steiner points to avoid complicated calculations in the path optimization process: 1) Delete other unnecessary Steiner points to facilitate the next step of Steiner point search once a feasible Steiner point is found; 2) Delete all the previous Steiner points to ensure the accuracy of the new path optimization after set the new interval value; 3) Delete all unnecessary Steiner points to ensure the smoothness of the path after completed the path optimization.

The flow chart of path optimization is shown in Figure 10.

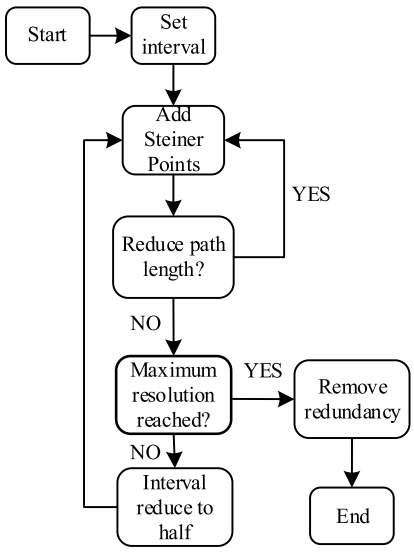

FIGURE 10. Fow chart of path optimization.

As shown in Figure 11, according to the rules of path optimization, V_1, V_3 are replaced, V_2, $V_4$are reserved. Other unnecessary Steiner points are deleted. The original path is $\{S \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow E\}$, and the optimized path is $\{S \rightarrow S_1 \rightarrow V_2 \rightarrow S_2 \rightarrow V_4 \rightarrow E\}$, and the path length has been significantly improved.

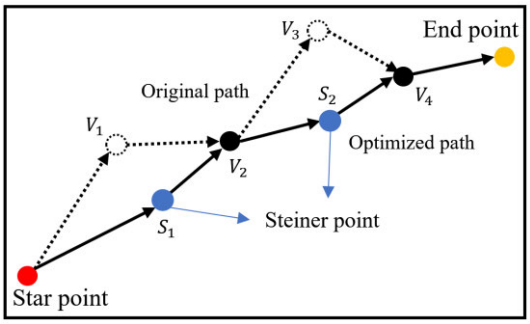

FIGURE 11. Path optimization.

IV. RESULT

A. EXPERIMENTAL CONDITION

To verify the rationality of the DFPA, we set up a 10 × 10 m mobile robot working range and conducted 750 random experiments.

The number of obstacles is 10, 10*10 cells, 150 experiments; the number of obstacles is 20, 10*10 cells, 150 experiments; the number of obstacles is 30, 10*10 cells, 150 experiments; the number of obstacles is 40, 20*20 cells, 150 experiments; the number of obstacles is 50, 20*20 cells, 150 experiments.

The coordinates and shapes of the obstacles were randomly generated, and the start and end points were the same.

The size of the mobile robot: L = 0.98m, W = 1 m.

The number of grids: Numgrid = 100.

The start point of the mobile robot was (0.5, 0.5) and the end point was (9.5, 9.5).

The cost consumption of the grid where the Voronoi point is absent was: field=ones (10,10) + rand (10,10)

The cost consumption of the grid where the Voronoi point was located was set to 1, which ensured that the Voronoi point was selected first. At the same time, the consumption values of other grids are different.

We set the moving boundary range of the mobile robot:

$$\begin{cases} X_{min} = 0m; & X_{max} = 10m \\ Y_{min} = 0m; & Y_{max} = 10m \end{cases} \tag{25}$$

We will discuss one of the 10 obstacle experiments. The coordinate points of obstacles are shown in Table 1.

The schematic diagram of obstacles is shown in Figure 12

B. CODE PROGRAM

The whole DFPA program consists of three parts: 1) Voronoi diagram generation, which generates Voronoi points according to the coordinates of obstacles. 2) Obstacle grid placement, which is divided by the main parameters of the mobile

TABLE 1. Obstacle coordinates.

Number	Coordinate (m)	
	X	Y
1	0.09	7.46
2	0.28	1.76
3	2.58	8.28
4	3.01	4.68
5	4.95	0.87
6	5.96	8.09
7	6.09	7.45
8	7.32	6.35
9	7.33	5.84
10	9.58	3.87

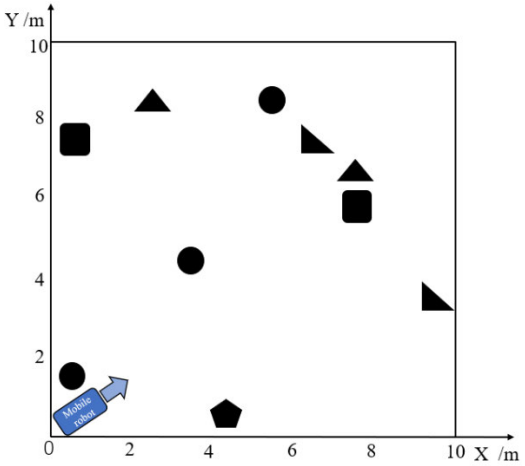

FIGURE 12. Distribution of obstacles.

robot and the map boundary grid, and then generates the obstacle grid according to the coordinates and shape of the obstacle, and stores the obstacle edges. 3) Pathfinding from the start point according to the algorithm rules; on the basis of not touching the obstacles, the path node is continuously searched until the end point is found, or if the end point cannot be found, the algorithm ends.

Algorithm 1 is based on the coordinates of the obstacle points to generate Delaunay triangles to further find all of the Voronoi points, and remove the Voronoi points beyond the boundary.

Generating the Voronoi points is the first step in map modeling, and also a key step in finding nodes. In the following program, the obstacle point coordinates also need to be used as an important reference for the obstacle visualization module.

Algorithm 1 generates usable Voronoi points. Algorithm 2 implements Obstacles grid placement. On the basis of early analysis, Algorithm 2 divides the number of grids, reflects the grid image occupied by obstacles, and expresses the Voronoi points on the map.

To allow the operator to check and simplify the obstacle edges, Algorithm 2 reflects all of the points and obstacle blocks in the map. The process of obstacle placement introduces the concept of the grid, which is equivalent to the

Algorithm 1 Voronoi Diagram Generation

Input:
Xdot ⟵ obstacle coordinate points
Xmin ⟵ Environmental boundary
Xmax ⟵ Environmental boundary
Ymin ⟵ Environmental boundary
Ymax ⟵ Environmental boundary
Output:
VP ⟵ Voronoi points within the boundary
Vpedge ⟵ Voronoi edges within the boundary
  For *Xdot* in every obstacle **do**
    If obstacle is within the Delaunay triangle **then**
      Trimat(i) ⟵ Delaunay triangle
    Else
      Trimat(i) ⟵ *f* (*xdot*) || Function connecting coordinate points
  Initialize *juedge* ⟵ 0
  For Every one of *Trimat* **do**
    Juedge ⟵ **Intersect**(*Trimat* (m), *Trimat* (m + 1))
  If *Juedge* >0 **then**
    voronoiedge ⟵ ***Maketricenter*** (*Trimat, Juedge*)
If *Xmin* < *voronoiedge*(1) < *Xmax* & Ymin < *voronoiedge*(2) < *Ymax* **then**
    VP ⟵ **Find**(*voronoiedge*)
    Vpedge ⟵ [*VP*(i),*VP*(i + 1)]
Algorithm End

Algorithm 2 Obstacles Grid Placement

Input:
Xdot ⟵ obstacle coordinate points
obs_shape ⟵ shape of the obstacle
Numgrid ⟵ Number of grids
Output:
obsedge ⟵ Edges of obstacles
Initialize *Mumgrid*
Initialize *Field*
If *obs_shape* =1 **then**
    obsxmin ⟵ **ceil**(*Xdot*) - *numl*
    obsxmax ⟵ **ceil**(*Xdot*) + *numr*
    obsymin ⟵ **ceil**(*Xdot*) - *numd*
    obsymax ⟵ **ceil**(*Xdot*) + *numu*
obsedge ⟵ **g**(*obsxmin, obsxmax, obsymin, obsymax*)
|| Function combined vertex coordinates
Algorithm End

introduction of the index value, and transforms coordinate points to index values. This is a key connection between early map modeling and subsequent pathfinding. In the pathfinding process, we mainly perform pathfinding based on coordinate points, but when the pathfinding enters a deadlock state, we must expand the pathfinding according to the grid.

Algorithm 1 and Algorithm 2 belong to the process of map modeling in the whole DFPA. After processing all elements,

the DFPA enters the pathfinding module of the algorithm, which is also the core part of the overall algorithm.

Algorithm 3 Pathfinding

Input:
Xdot ⟵ obstacle coordinate points
obsedge ⟵ shape of the obstacle
Numgrid ⟵ Number of grids
VP ⟵ Voronoi points within the boundary
S ⟵ Start point
E ⟵ End point
Output:
path
Initialize close ⟵ [ ]
Initialize open ⟵ [*S*]
While *E* is not *open* && open is not empty **do**
  P ⟵ **Find min**(*open*)
  If *P* cannot be directly connected to *E* **then**
  NewVP ⟵ **FindRecnet**(*P,VP*)
  || Function find the nearest *VP*
    If *P* can be directly connected to *VP* **then**
      open ⟵ *NewVP*
    Else
    Newpoint ⟵ **A-star**(*P*)
    || Function A-star extended pathfinding
    open ⟵ *Newpoint*
  Else
  open ⟵ [*P,E*]
If open contains *E* **then**
  Path ⟵ *close* || Solution found
  NewPath ⟵ **Optimization**(*Path*) || Steiner point
Else
  Path ⟵ 0 || No solution
Algorithm End

C. DYNAMIC FUSION PATHFINDING ALGORITHM RESULT

Under the experimental conditions shown in Fig.12, the entire process of a mobile robot using the DFPA for path planning, which is shown in Figure 13

First, we determine the geometric center point of the obstacles and use the Delaunay triangulation theory to generate the Delaunay triangle grid and Voronoi diagram, as shown in Fig.13-a.

Next, we divide the grid according to obstacles, place the obstacles within the grid, and extract all obstacle edges, as shown in Fig.13-b.

The third step is to determine the pathfinding of the start and end points. It is judged whether the start point can be directly connected to the end point without intersecting the obstacle edges and whether it can be connected to the nearest Voronoi point. If this is not possible, an extended road search is required, as shown in Fig.13-c.

When the current path node cannot reach the end point and the nearest Voronoi point directly, the mobile robot selects the neighboring nodes with the least cost as the new path node; as shown in Fig.13-d, the six purple nodes points are candidate nodes for extended pathfinding, and yellow nodes are the least costly nodes.

The new path node continues to be judged with the end point, and the robot looks for the next path node, as shown in Fig.13-e.

Finally, the entire path diagram is obtained, the optimized path is shown in Fig.13-f.

Fig.13 clearly describes the steps of the entire method, we found a feasible path based on DFPA under this experimental condition. A total of nine usable Voronoi points was generated during the map modeling process. During the pathfinding process, although Voronoi points were considered as priority path nodes, only three Voronoi points were used as path nodes. According to the pathfinding rules, the improved A* algorithm was used to expand the search for three nodes. The entire path had a total of six path nodes, which realizes the path search of the mobile robot from the start point to the end point.

However, this experimental condition is relatively simple and cannot fully prove the feasibility of DPFA. Therefore, we chose four complex experimental conditions for discussion. The feasible paths of these four experiments are shown in Figure 14.

We can see that all four experiments have obtained a feasible path. The path nodes in Fig.14-a and Fig.14-b are mainly Voronoi points. As the number of obstacles increases, the path becomes more complicated. As shown in Fig.14-c, it is impossible to find a feasible path in a more complex environment using only the Voronoi point. Therefore, three deadlocks occurred. It took three additional path nodes to find the path. In the case of 50 obstacles, the end point is almost surrounded by obstacles, as shown in Fig.14-d. However, we can find that DFPA can still find a feasible path. Expanding the search for path nodes makes the robot jump out of the dire situation of being surrounded by obstacles.

Summarizing the results of these 750 experiments, we can find some characteristics of DFPA. The results of all simulation experiments are shown in Table 2.

where *Exps* is the number of experiments, *Obs* is the number of obstacles, *Atime* is the average calculation time of each experiment, *Nos* is the number of experiments where no feasible solution could be obtained, *Spath* is the number of feasible paths obtained by each group of experiments, and *Srate* is the success rate of obtaining feasible paths.

When the number of obstacles was 10 or 20, the success rate was 100%. As the number of obstacles increased, the success rate decreased. When the number of obstacles was 30, 147 paths were obtained; when the number of obstacles was 40, 145 paths were obtained; when the number of obstacles was 50, 144 paths were obtained. The algorithm operated at high speed, but as the number of obstacles increased, the calculation speed of the algorithm also increased.

As shown in Fig.15, when there were 10 obstacles, the average algorithm time was 0.0132s. When there were 50 obstacles, the average algorithm time reached 0.1563s.

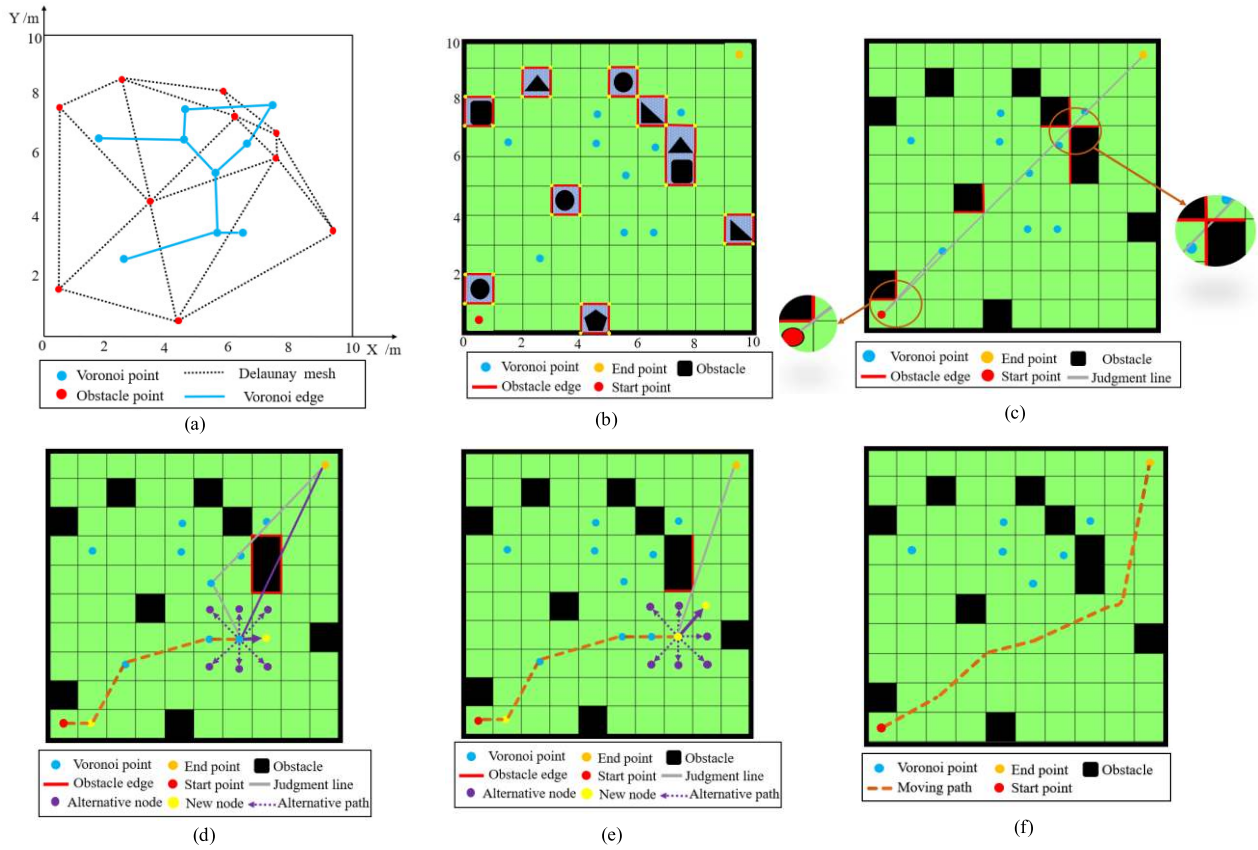

FIGURE 13. Process of the DFPA: (a) Delaunay triangle grid and Voronoi diagram; (b) all obstacle edges; (c) the judgment of whether the start point can be directly connected to the end point; (d) extended search path node; (e) continue to find path nodes; (f) overall movement path.

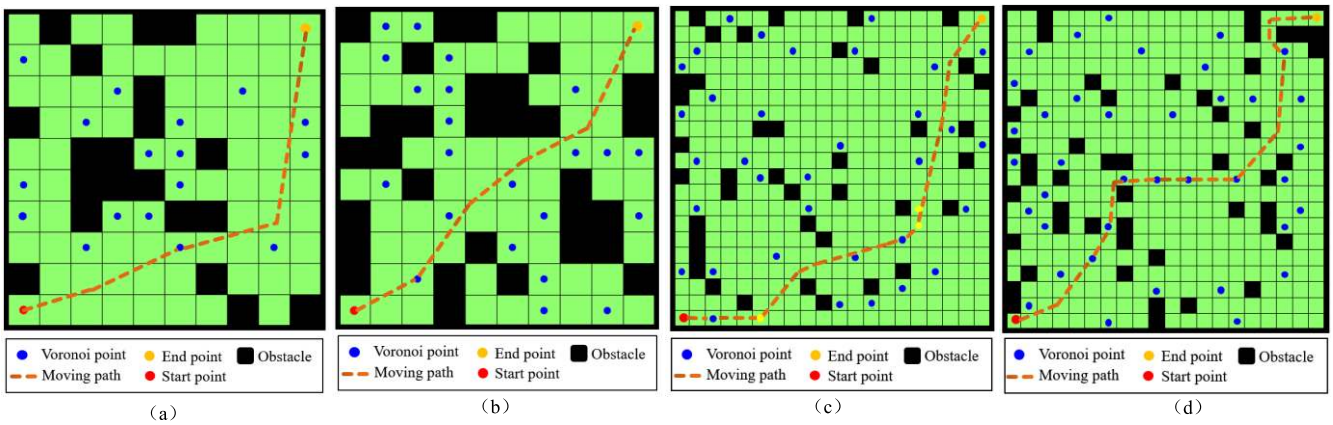

FIGURE 14. Feasible paths obtained through the DPFA in different environments: (a) obstacle number=20; (b) obstacle number=30; (c) obstacle number=40; (d) obstacle number=50.

TABLE 2. DFPA results for 750 MAPS with different environmental.

Number	Exps	Obs	Atime(s)	Nos	Spath	Srate
1	150	10	0.0132	0	150	100%
2	150	20	0.0265	0	150	100%
3	150	30	0.0869	3	147	98%
4	150	40	0.1136	5	145	96.67%
5	150	50	0.1563	6	144	96%

D. COMPARISON OF PATH PLANNING METHODS

To evaluate the performance of the DFPA, we compared it with other related methods, i including the A*, RRT, GA and ACO algorithms. First, we must ensure the consistency of a series of prerequisite elements such as number and location of obstacles, start point, and end point to ensure the credibility of the comparison experiment results.

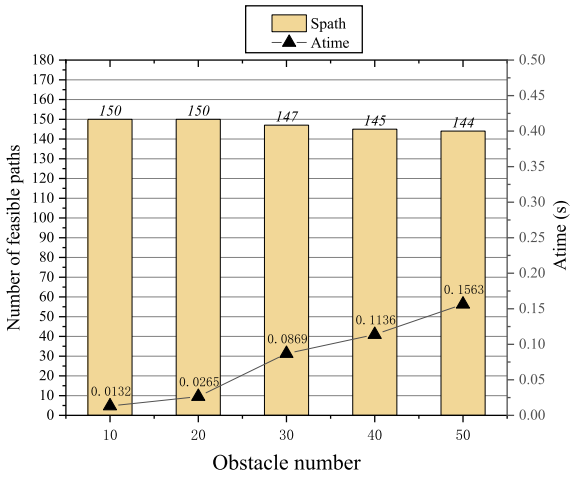

FIGURE 15. Average algorithm time and the number of paths obtained by the DFPA.

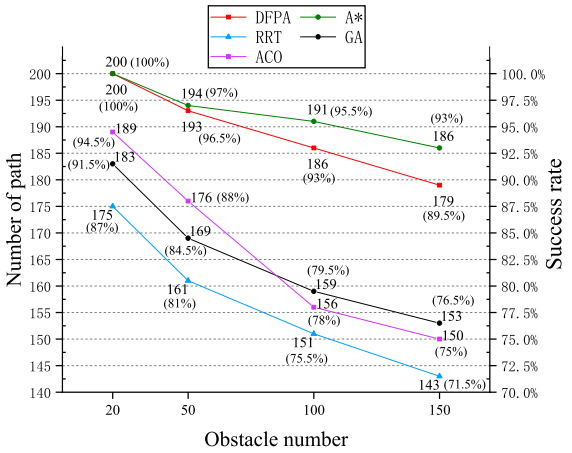

FIGURE 16. The number of feasible paths with the number of obstacles is 20, 50, 100,150.

In reality, each turning movement of the mobile robot takes a certain amount of time. Therefore, the time for the mobile robot to travel from the start point to the end point at the same speed depends on both the length of the planned path and the number of turns. The smaller the number of turns and the smaller the turning angle, the more efficient the robot's operation. To this end, the turning angle and turning times were introduced, and Equation (26) for defining the turning consumption cost is given as follows.

$$Tconsum = \sum_{i=1}^{n} (\theta_i) \tag{26}$$

where n is the number of turns, θ_i is the radian of the i-th turn, and the total consumption cost of the robot can be calculated by the number of turns and the angle of each turn.

The simulation experiment was carried out in an environment where the map range was 10 × 10 m, and the number of obstacles was 20, 50, 100, and 150. There are 200 maps under each obstacle quantity and a total of 800 experimental maps. We performed the simulation using MATLAB 2016a, with computer memory of 8 GB. The final comparison results are shown in Table 3:

where: *Maps* is the number of the experimental maps, *Alength* is the average length of the path, and *ATconsum* is the average turning cost of the path.

The comparing of the DFPA with the A* and RRT algorithms used analysis of four aspects: success rate, algorithm running time, path length, and turn consumption.

1) SUCCESS RATE

In the experimental environment of 20, 50, 100, and 150 obstacles, 800 experiments were carried out on DFPA, A*, RRT, GA and ACO, and the number of feasible paths and success rate obtained is shown in Figure 16.

With the increase of obstacles, the number of valid paths found by the five methods showed a downward trend, but it can be seen from several comparative experiments that the success rate of the A* is the highest. The A* can search the entire map, the ability to find a path was the strongest. In an environment of 20 obstacles, the DFPA also obtained 200 feasible paths with a success rate of 100%. The DFPA and A* had the same success rate. However, in an environment of 150 obstacles, the DFPA obtained 179 possible paths, and the success rate dropped to 89.5%, which was less than 93% of the A*. As the number of obstacles increases, the DFPA will no longer have the advantage, because DFPA used the Voronoi point as the priority pathfinding node. The increase in the number of obstacles will result in the reduction of the available Voronoi points. In the case of a large number of obstacles, the pathfinding ability of DFPA is not as good as A*, but it is far better than other methods.

2) ALGORITHM TIME

Algorithm running time is an important indicator to measure the performance of the DFPA, and running time can measure the implementation of path planning by the DFPA for mobile robots. In each experiment, we obtained the running time of each method, as shown in Figure 17; Fig. 17-a is a box-plot of the A* algorithm running time, Fig.17-b is a box-plot of the RRT algorithm running time, Fig.17-c is a box-plot of the GA running time, Fig.17-d is a box-plot of the ACO algorithm running time, and Fig.17-e is a box-plot of the running time of the DFPA algorithm.

In the environment with 20 obstacles, the average calculation time of the DFPA was 0.02873 s, in the environment with 50 obstacles, the average calculation time of the DFPA was 0.17172 s, in the environment with 100 obstacles, the average calculation time of the DFPA was 0.7359 s, and in the environment with 150 obstacles, the average calculation time of the DFPA was 1.75 s. As the number of obstacles increased, the running time of the DFPA increased. The running times of other algorithms showed a similar pattern. In the case of 20 obstacles, the running time of DFPA was relatively balanced. Furthermore, Figure 17-e shows the height of the box plot is small, while the height of the A* and RRT time box plots is large, the height of the GA and ACO time box plots is larger, and the time dispersion is high, indicating that

TABLE 3. Comparison results.

	Maps	Obs	Atime (s)	Nos	Spath	Srate	Alength (m)	ATconsum
DFPA	200	20	0.03	0	200	100.00%	13.75	0.68
	200	50	0.17	7	193	96.5%	15.08	1.20
	200	100	0.74	14	186	93.00%	17.26	1.95
	200	150	1.75	21	179	89.50%	18.98	2.32
A*	200	20	1.16	0	200	100.00%	20.51	3.87
	200	50	1.23	6	194	97.00%	22.20	11.58
	200	100	1.25	9	191	95.50%	25.09	17.19
	200	150	1.26	14	186	93.00%	27.22	24.96
RRT	200	20	0.29	25	175	87.00%	20.40	25.59
	200	50	0.46	39	161	81.00%	22.87	28.05
	200	100	0.63	49	151	75.50%	24.51	34.02
	200	150	0.71	57	143	71.50%	25.91	40.56
GA	200	20	1.63	17	183	91.50%	20.51	9.75
	200	50	1.69	31	169	84.50%	22.20	22.02
	200	100	1.92	41	159	79.50%	25.10	31.08
	200	150	1.93	47	153	76.50%	27.22	42.63
ACO	200	20	1.71	11	189	94.50%	20.40	11.67
	200	50	1.71	24	176	88.00%	22.87	18.24
	200	100	1.90	44	156	78.00%	24.51	32.46
	200	150	2.04	50	150	75.00%	25.91	44.67

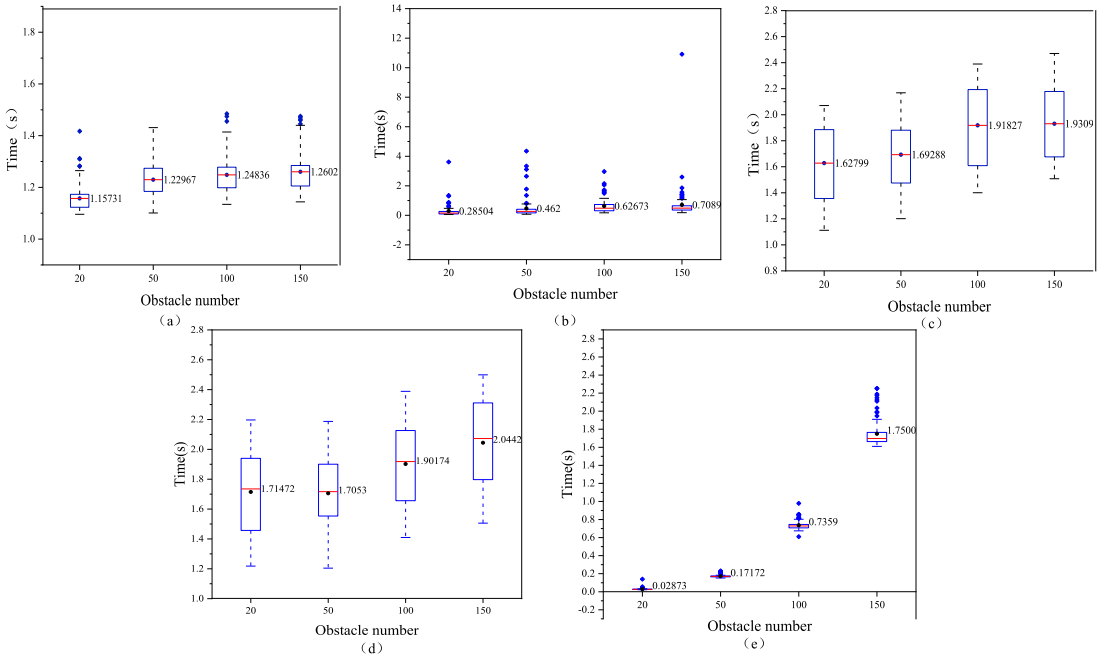

FIGURE 17. Different algorithm running times under 20, 50, 100, and 150 obstacles: (a) A*; (b) Rapidly-exploring Random Tree (RRT); (c) GA; (d) ACO; (e) DFPA.

the DFPA in the case of 20 obstacles ran smoothly. However, as the number of obstacles increased, the dispersion of DFPA running time also increased, and the stability continued to decline.

By simulating DFPA, A*, RRT, GA and ACO in the same environment with 0∼150 obstacles, the running time comparison of the three methods under each obstacle environment was obtained, as shown in Figure 18.

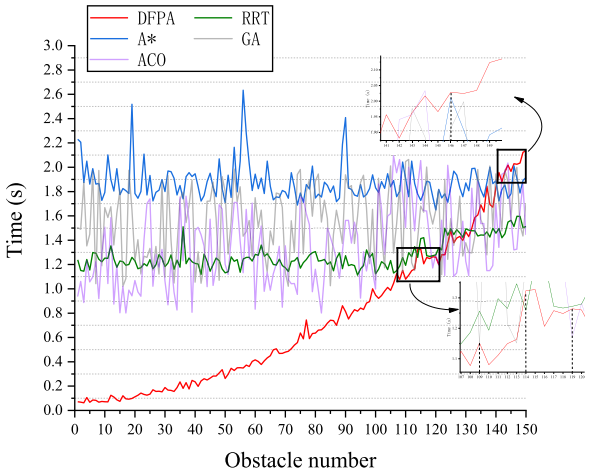

FIGURE 18. The running time of DFPA, A*, RRT, GA, and ACO from 0 to 150 obstacles.

The running time of A* and RRT algorithms fluctuated little and was less affected by the number of obstacles. However, the running time of GA and ACO algorithms fluctuated greatly. The running time of DFPA increased as the number of obstacles increased. When the number of obstacles was less than 109, the running time of the DFPA was much shorter than that of A*, RRT, GA, and ACO. When the number of obstacles was higher than 146, the running time of DFPA was higher than those of A*, RRT, GA, and ACO. The DFPA is most affected by obstacles. As the number of obstacles increases, the running time of DFPA will far exceed other methods. The main reason is that DFPA generates Voronoi points and obstacle edges based on obstacle coordinate points when path planning. The increase in the number of obstacles increases the processing time for the obstacles. However, in the actual operating conditions of the mobile robot, due to the large size of the mobile robot and the small distance between obstacles, different obstacles that are closer together will be placed in an obstacle grid so that the calculated number of obstacles is often less than the actual number, which does not affect the operating speed of the DFPA.

3) PATH LENGTH

The length of the path obtained in each experiment is shown in Figure 19. Fig.19-a is a scatter plot of the A* path length, Fig.19-b is a scatter plot of the path length of the RRT algorithm, Fig.19- c is a scatter plot of the path length of the GA, Fig.19-d is a scatter plot of the ACO path length, Fig.19- e is a scatter plot of the path length of the DFPA.

In the environment of 20 obstacles, 200 experiments were carried out, and DFPA obtained 200 possible paths. Among these, paths with a length of 14.007 m were reached 153 times, which means that the mobile robot reached the end point directly from the start point, and the shortest path planning was realized. As the number of obstacles increased, the path became more complicated, and the length increased, but the path length of the DFPA was shorter than that of the RRT, A*, GA, and ACO under the same number of obstacles. The length of the feasible path obtained by the five algorithms in different obstacle environments is shown in Figure 20.

In the case of a small number of obstacles, the most path length of the A* algorithm was 20m. When the number of obstacles was large, the path of the A* algorithm will be more complicated, and the path length will be longer. In the

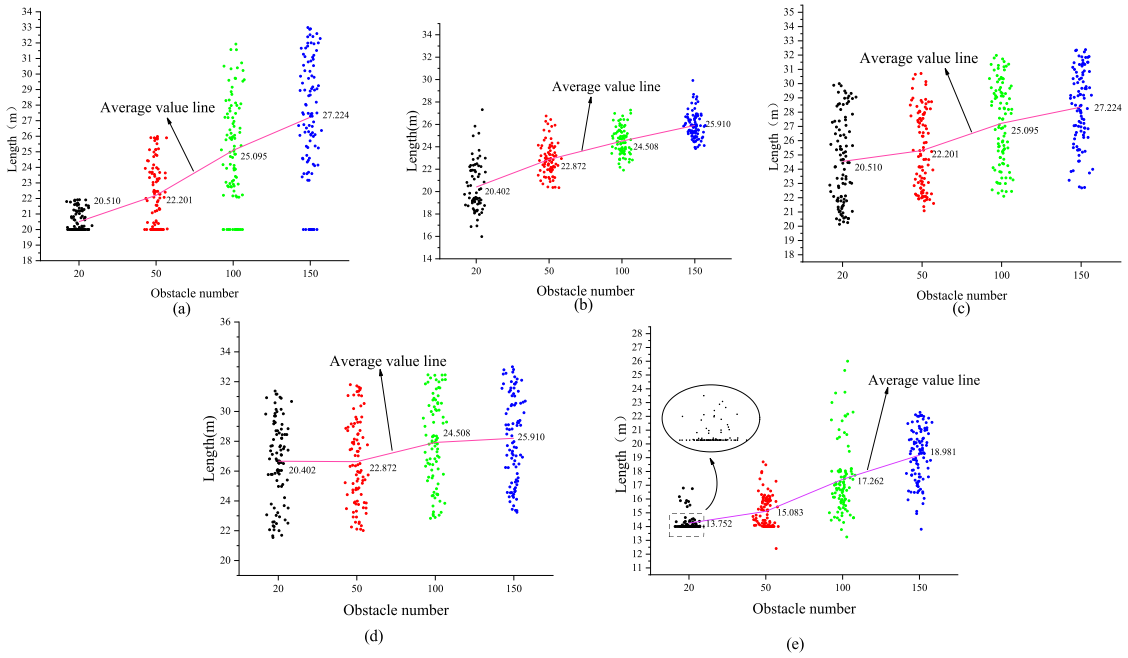

FIGURE 19. Path length under 20, 50, 100, and 150 obstacles: (a) A*; (b) RRT; (c) GA; (d) ACO; (e) DFPA.

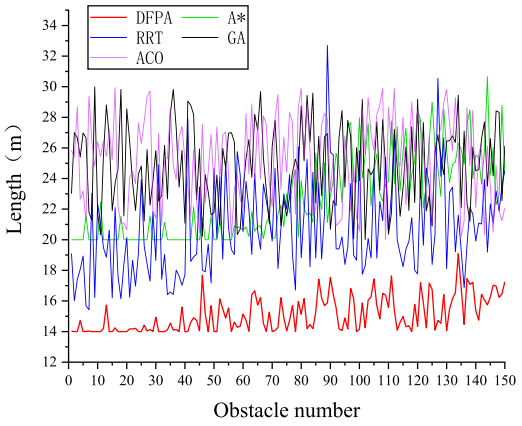

FIGURE 20. The path length of DFPA, A*, RRT, GA, and ACO from 0 to 150 obstacles.

same obstacle environment, the path length of the DFPA was always lower than those of A*, RRT, GA, and ACO.

4) TURNING COSTS

Turning cost is an essential reference for evaluating the path of a mobile robot. In the actual working conditions of mobile robots, the primary energy consumption is that of turning consumption, especially in a complex obstacle environment. An excessive number of turns and large-angle turns will lead to low efficiency of the mobile robot. When the mobile robot can reach the end point directly from the starting point, without turning during this period, the cost of turning is 0. The 200 turning cost data for four experimental environments with 20, 50, 100, and 150 obstacles are plotted as a scatter plot, as shown in Fig.21-a. In the case of 20 obstacles, the turning cost distribution is shown in Fig.21-b.

In the case of 20 obstacles, most of the data is concentrated at 0, which means that most paths can reach the end point without turning to avoid obstacles. As the number of obstacles increases, the edges of obstacles increase, the path becomes more complicated, and there are fewer cases of reaching the end point from the start point without turning. A comparison of the turning costs of DFPA, A*, RRT, GA, and ACO under different obstacles, is shown in Figure 22.

The turning cost of the RRT algorithm under different obstacles was the highest. As the number of obstacles increased, the turning cost of the A*, GA, and ACO algorithm increased. The DFPA was always at a low level, and far lower than those of A*, RRT, GA, and ACO. In the process of path node selection, the DFPA can search the Voronoi point first and determine whether it can reach the end point directly, which achieves the least number of node selections and enables the mobile robot to achieve the least number of turns.

Through simulation experiments, we compared the DFPA with A*, RRT, GA, and ACO. The results show that DFPA has better performance in some aspects. The DFPA has a higher success rate in obtaining a feasible path. In the case of fewer obstacles, the algorithm running time is also lower than that those of A*, RRT, GA, and ACO. Under different obstacle environments, path lengths, and turn consumption, the DFPA has a significant advantage.

E. PERFORMANCE SUPERIORITY OF THE DFPA

We have compared DFPA with four methods. It can be concluded that DFPA has advantages in success rate and turn consumption, but the operating speed is insufficient. To find out which obstacle type performs best for DFPA, we refer to the existing instances in [36]and compare DFPA with the other methods again to further verify the superiority of DFPA and find the best environment for DFPA. The size of the entire map is 10m*10m, and is divided into 10*10 cells. To ensure the correctness of the path trajectory, the blue path of GA in Fig.23-a is derived from Figure12-a of [36], and the yellow path of A*\ACO in Fig.23-a is derived from Figure12-c of [36]. The blue path in Fig.23-b comes from Figure13-g

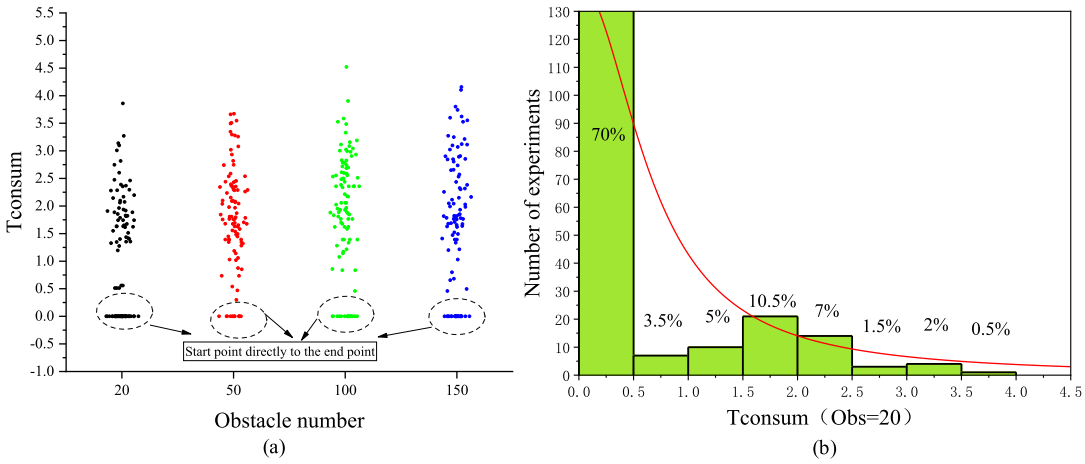

FIGURE 21. Turning cost of the DFPA: (a) the scatter plot of turning costunder 20, 50, 100, and 150 obstacles; (b) the turning cost distribution under 20 obstacle.

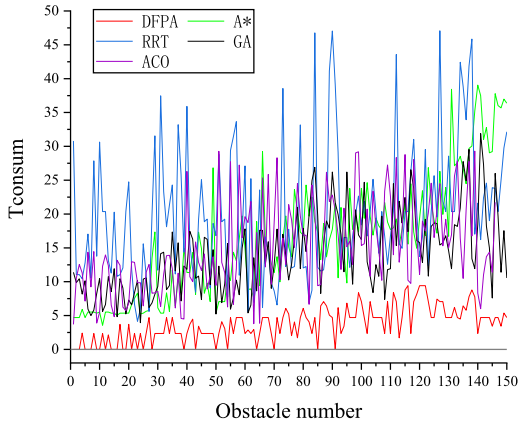

FIGURE 22. Turning costs of DFPA, A*, RRT, GA, and ACO from 0 to 150 obstacles.

in [36]. The red line is the path trajectory generated by the DFPA. The comparison path trajectory is shown in Fig.23.

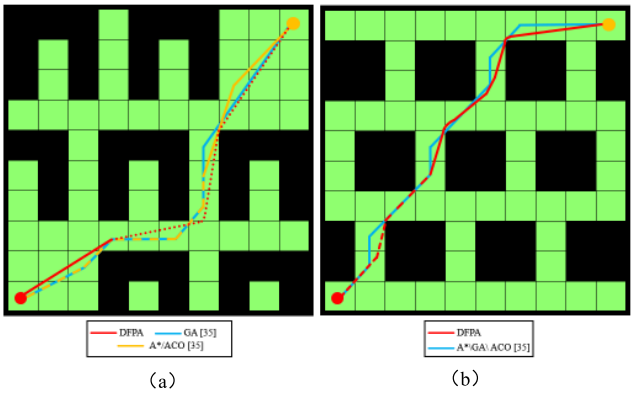

FIGURE 23. Path trajectory comparison of DFPA and [35].

Similarly, we compare the DFPA with the [37], the black path of RRT* in Figure 24 is derived from Figure14-b of [37].

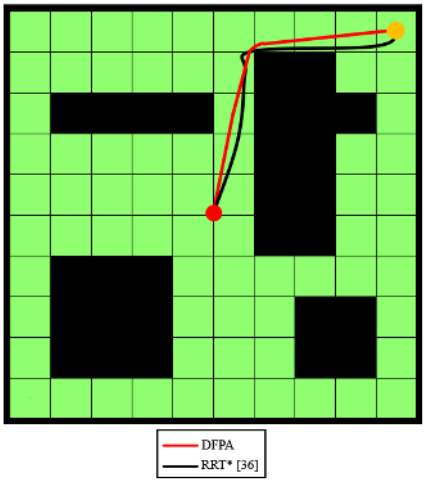

FIGURE 24. Path trajectory comparison of DFPA and [36].

The red line is the path trajectory generated by the DFPA. The comparison path trajectory is shown in Fig. 24.

We summarized the length and turn consume of all paths into Table 4.

TABLE 4. Comparison results.

Maps	Methods	Length（m）	Tconsum
Fig.23-a	DFPA	14.449	1.836
	GA	14.366	3.351
	A* \ACO	14.362	3.490
Fig.23-b	DFPA	13.632	4.762
	A*\GA\ACO	14.451	5.463
Fig.24	DFPA	8.654	1.359
	RRT	8.950	6.349

In the N-type obstacle environment (Fig.23-a), the path length of DFPA is 14.449m, the path length of GA is 14.366m, and the path length of A*\ACO is 14.362m. DFPA has no advantage. This is due to the N-shape Obstacles will cause the Voronoi point to be in the N-shaped slot, which is difficult to use as a priority pathfinding node. In Fig.23-b, the path length of DFPA is 13.632m, which is significantly better than the 14.451m of A*\GA\ACO. However, the turning consumption of DFPA is significantly smaller than other methods. In Fig.24, DFPA shows good performance in this environment without N-shaped obstacles. It can be seen that the path of DFPA is very flexible, the regular rectangular obstacles produce a sufficient number of Voronoi points which provide flexible paths. DFPA performs poorly in N-type obstacles, but has obvious advantages in rectangular obstacle environments.

After experiments, we found that DFPA is significantly improved compared to A*, and is also better than RRT, GA, and ACO in running time, path length, and turn consumption. DFPA performs well in rectangular obstacle environments. The DFPA adopts a priority to judge whether the destination can be reached directly, which can simplify the path node search, realize the shortest path length, use the Voronoi point as the priority pathfinding node, realize the minimum turning cost, and dynamically combine obstacle edges and pathfinding strategies. The performance of the DFPA was significantly improved, and the experimental results proved that the DFPA achieved better results.

V. DISCUSSION

Through algorithm comparison and experiments, it can be seen that the DFPA can realize the map construction and pathfinding of mobile robots. Compared with the A*, RRT, GA, and ACO algorithms, the DFPA algorithm obtains fewer path nodes, significantly reduces path length and turning consumption, has a high success rate in obtaining a possible path, and has a short calculation time. The priority of the Voronoi point realizes the path simplification of nodes and grid filling reduces the number of obstacles gathered. Direct judgment and new pathfinding processes reduce the number

of iterations. Obstacle edges are used as the basis for judging intersections and collisions, simplifying the algorithm's complexity. New map modeling methods and improved A* makes the path more direct and reduces the number of turns, so the path length and turn consumption costs were significantly reduced. In a simple experimental environment, path length and turning consumption are very low, which means that the mobile robot directly reached the end point from the start point. In a complex experimental environment, DFPA has advantages in turning advantages and path length, which showed good advantages. Our research results are consistent with the related research on path planning of mobile robots [15], [19], [26]. For example, Fu *et al.* [26] used the A* algorithm to propose a pre-judgment from the start point to the end point, which improved the path success rate and shortened the path length by 20% to 66%;however, this approach did not introduce a method of avoiding obstacles. Ayawli *et al.* [19] used the Voronoi diagram to find the initial path, and then classify and place the obstacles. Replanning the path allowed the robot to avoid obstacles flexibly, thereby improving the path length and success rate. However, repeated path planning leads to an increase of calculation time. The method proposed in the current paper realizes dynamic fusion map construction and pathfinding, provides an effective path planning method for mobile robots.

The method proposed in this paper applies the Voronoi diagram to the path planning of mobile robots, simplifies the path node of the A* algorithm, and expands the use of Voronoi diagrams. Furthermore, the pathfinding approach has reference significance for the planning of large outdoor sites. Obstacle placement and the approach of determining the side of the obstacle provide the basis for the obstacle avoidance method of automatic navigation. However, the DFPA also has shortcomings. When there are many obstacles, this approach takes more time to process obstacles and generate Voronoi points. According to the distance between obstacles and the size of the mobile robot, the number of obstacles can be combined to obtain a minimum. The calculation complexity of the Delaunay triangulation algorithm affects the calculation time of the algorithm, and related improvement work needs to be continued. The DFPA only considers static obstacles during verification and does not consider moving obstacles. Due to the influence of moving direction, speed, and floor area, dynamic obstacles can be measured and judged by many sensors. Based on the research ideas in this paper, a reasonable mathematical model can be established and added to the overall algorithm process. The fast extraction of location coordinate points and obstacle edges is the next research direction.

VI. CONCLUSION

In this paper, the Delaunay triangulation algorithm is used to integrate disordered obstacles into an ordered topological regional relationship, and the grid is introduced to process obstacles to identify obstacle shapes and extract obstacle edges. Based on the A* algorithm, direct judgment, Voronoi point priority pathfinding and traditional pathfinding are combined to design a new pathfinding process. The obstacle edges of the map modeling are integrated to achieve obstacle avoidance and path planning for mobile robots.

The main findings are as follows:

1) The reduction of path nodes reduces the number of turns and associated turn cost. As the number of obstacles increases, the cost of turning is far less than those of the A*, RRT, GA, and ACO. The reduction of path nodes also makes the path simpler, and the path length is greatly reduced. The more direct path length increases the movement efficiency of the mobile robot.

2) Obstacle placement can reduce the number of obstacles and improves the algorithm speed in the case of a large number of obstacles. Through the DFPA, the success rate of pathfinding significantly improved; with less than 150 obstacles, the success rate of pathfinding is above 90%.

3) The DFPA has good advantages for rectangular obstacle environments, especially low turning consumption and path length.

REFERENCES

[1] 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.

[2] M. Dakulović and I. Petrović, "Two-way D* algorithm for path planning and replanning," *Robot. Auton. Syst.*, vol. 59, no. 5, pp. 329–342, May 2011.

[3] M. A. P. Garcia, O. Montiel, O. Castillo, R. Sepulveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1102–1110, Jun. 2009.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] W.-B. Xu, X.-B. Chen, J. Zhao, and X.-P. Liu, "Function-segment artificial moment method for sensor-based path planning of single robot in complex environments," *Inf. Sci.*, vol. 280, pp. 64–81, Oct. 2014.

[10] 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.

[11] T. Gawron and M. Michałek, "A G3-continuous extend procedure for path planning of mobile robots with limited motion curvature and state constraints," *Appl. Sci.*, vol. 8, no. 11, p. 2127, Nov. 2018.

[12] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput.*, vol. 77, pp. 236–251, Apr. 2019.

[13] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5177–5191, 2015.

[14] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: Electrostatic potential field approach," *Expert Syst. Appl.*, vol. 100, pp. 68–78, Jun. 2018.
[15] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, pp. 257–266, Dec. 2015.
[16] L. Perumal, "New approaches for delaunay triangulation and optimisation," *Heliyon*, vol. 5, no. 8, Aug. 2019, Art. no. e02319.
[17] Z. Yeeh, Y. Song, J. Byun, S.-J. Seol, and K.-Y. Kim, "Regularization of multidimensional sparse seismic data using delaunay tessellation," *J. Appl. Geophys.*, vol. 174, Mar. 2020, Art. no. 103877.
[18] F. Yu, Y. Zeng, Z. Q. Guan, and S. H. Lo, "A robust delaunay-AFT based parallel method for the generation of large-scale fully constrained meshes," *Comput. Struct.*, vol. 228, Feb. 2020, Art. no. 106170.
[19] B. B. K. Ayawli, X. Mei, M. Shen, A. Y. Appiah, and F. Kyeremeh, "Mobile robot path planning in dynamic environment using Voronoi diagram and computation geometry technique," *IEEE Access*, vol. 7, pp. 86026–86040, 2019.
[20] M. Candeloro, A. M. Lekkas, and A. J. Sørensen, "A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels," *Control Eng. Pract.*, vol. 61, pp. 41–54, Apr. 2017.
[21] Q. Wang, M. Langerwisch, and B. Wagner, "Wide range global path planning for a large number of networked mobile robots based on generalized Voronoi diagrams," *IFAC Proc. Vols.*, vol. 46, no. 29, pp. 107–112, 2013.
[22] 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.
[23] X. Zhang, Y. Zhao, N. Deng, and K. Guo, "Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm," *Int. J. Adv. Robotic Syst.*, vol. 13, no. 3, p. 91, Jun. 2016.
[24] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.
[25] W. Yin and X. Yang, "A totally astar-based multi-path algorithm for the recognition of reasonable route sets in vehicle navigation systems," *Procedia-Social Behav. Sci.*, vol. 96, pp. 1069–1078, Nov. 2013.
[26] 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.
[27] R. Song, Y. Liu, and R. Bucknall, "Smoothed A* algorithm for practical unmanned surface vehicle path planning," *Appl. Ocean Res.*, vol. 83, pp. 9–20, Feb. 2019.
[28] P. G. Luan and N. T. Thinh, "Real-time hybrid navigation system-based path planning and obstacle avoidance for mobile robots," *Appl. Sci.*, vol. 10, no. 10, p. 3355, 2020.
[29] B. K. Patle, A. Pandey, A. Jagadeesh, and D. R. Parhi, "Path planning in uncertain environment by using firefly algorithm," *Defence Technol.*, vol. 14, no. 6, pp. 691–701, Dec. 2018.
[30] M. A. Hossain and I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Robot. Auto. Syst.*, vol. 64, pp. 137–141, Feb. 2015.
[31] C. Kim, Y. Kim, and H. Yi, "Fuzzy analytic hierarchy process-based mobile robot path planning," *Electronics*, vol. 9, no. 2, p. 290, Feb. 2020.
[32] M.-C. Rivara and J. Diaz, "Terminal triangles centroid algorithms for quality delaunay triangulation," *Comput.-Aided Des.*, vol. 125, Aug. 2020, Art. no. 102870.
[33] D. Contreras and N. Hitschfeld-Kahler, "Generation of polyhedral delaunay meshes," *Procedia Eng.*, vol. 82, pp. 291–300, Jan. 2014.
[34] X. Li, A. Krishnamurthy, I. Hanniel, and S. Mcmains, "Edge topology construction of Voronoi diagrams of spheres in non-general position," *Comput. Graph.*, vol. 82, pp. 332–342, Aug. 2019.
[35] P. Bhattacharya and M. Gavrilova, "Roadmap-based path planning–using the Voronoi diagram for a clearance-based shortest path," *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, Jun. 2008.
[36] H. Shin and J. Chae, "A performance review of collision-free path planning algorithms," *Electronics*, vol. 9, no. 2, p. 316, 2020.
[37] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

ZHIHAI LIU received the B.E. degree in mechanical and electronic engineering from the Shandong Mining Institute, China, in 1997, and the M.E. degree in mechanical design and theory and the Ph.D. degree in mechanical and electronic engineering from the Shandong University of Science and Technology, China, in 2000 and 2015, respectively. He is currently an Associate Professor with the College of Transportation, Shandong University of Science and Technology. His current research interests include condition monitoring of electrical machines, automatic control, and robotics.

HANBIN LIU was born in 1996. He received the B.E. degree in logistic engineering from the Shandong University of Science and Technology, China, in 2018. His research interests include mobile robot path planning, intelligent logistics, and industrial automation.

ZHENGUO LU received the Ph.D. degree in machine design and theory from the Shandong University of Science and Technology, in 2018. He is currently an Associate Professor with the Shandong University of Science and Technology and also with Shandong Normal University. He has published four articles, as a Principle Person. His research interests include underground coal mining and mine electromechanical. He has participated in more than ten projects funded by the Natural Science Foundation of China and the Natural Science Foundation of Shandong Province, China.

QINGLIANG ZENG received the Ph.D. degree in machine design and theory from the China University of Mining and Technology, in 2000. He is currently a Professor with the Shandong University of Science and Technology and also with Shandong Normal University. He has published more than 90 articles, as a Principle Person. His research interests include electromechanical integration, condition monitoring and fault diagnosis, and virtual prototype. He has participated in more than 40 projects funded by the National Sci-Tech Support Plan, National 863 Program, and the Natural Science Foundation of China.