IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received July 28, 2020, accepted August 8, 2020, date of publication August 12, 2020, date of current version August 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3015976

Surface Optimal Path Planning Using an Extended Dijkstra Algorithm

MIN LUO[1,2], XIAORONG HOU[1], AND JING YANG[1]
[1]School of Automation Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]School of Electrical Engineering and Information, Southwest Petroleum University, Chengdu 610500, China

Corresponding author: Xiaorong Hou (houxr@uestc.edu.cn)

ABSTRACT Extensive studies have been conducted on the Dijkstra algorithm owing to its bright prospect. However, few of them have studied the surface path planning of mobile robots. Currently, some application fields (e.g., wild ground, planet ground, and game scene) need to solve the optimal surface path. This paper proposes an extended Dijkstra algorithm. We utilize the Delaunay triangulation to model the surface environment. Based on keeping the triangle side length unchanged, the triangle mesh on the surface is equivalently converted into a triangle on the two-dimensional plane. Through this transformation, we set up the two-dimensional developable passable channel of the surface and solve the optimal route on this channel. Traversing all the two-dimensional developable and passable paths of the surface, we can get the shortest route among all the optimal paths. Then the inverse transformation from the two-dimensional plane coordinates to the corresponding surface coordinates obtains the surface optimal path. The simulation results show that, compared with the traditional Dijkstra algorithm, this method improves the accuracy of the surface optimization path in single-robot single-target and multi-robot multi-target path planning tasks.

INDEX TERMS Dijkstra algorithm, path planning, surface, Delaunay triangulation, mobile robots, optimization methods.

I. INTRODUCTION

Dijkstra algorithm is a classical well-known shortest path routing algorithm in 2D mobile robots' path planning researches. It is a simple algorithm for the single-source shortest path problem, which can effectively calculate the shortest path to all destinations [1]–[4].

The Dijkstra algorithm was introduced by Dutch computer scientist Edsger Wybe Dijkstra in 1959. It has been successfully applied in fields like mobile robot 2D path planning, computer science, geographic information science, and transportation, etc. Some recent research based on the Dijkstra algorithm is shown below [5]–[18].

Wolfgang Fink *et al.* adopted a multi-objective variant of the Dijkstra algorithm based on terrain data to achieve the overall optimal traversal in the 3D surface (2019) [5]. The gained results were employed in the Global Rover Horizontal Optimization Planner (GRTOP) automation system to quickly and accurately set up optimized routes for multiple constraints at the same time. This research enabled GRTOP to reprogram traversal/task frequently, and optimized traversal and task security. The authors exploited the diamond square algorithm to create realistic terrain in the surface environment. The weight between two consecutive points was taken as the Euclidean distance between them. The Dijkstra algorithm was extended to consider multiple targets, and the weights between adjacent nodes were set as a linear combination of multiple weights. Each weight corresponded to a specific target. Through these methods, the Dijkstra algorithm was applied to the calculation of a three-dimensional optimal path and could be extended to multi-objective tasks.

Dong Guo *et al.* improved the traditional Dijkstra algorithm and combined it with the vehicle fuel consumption and emission measurement model to reduce vehicle fuel consumption and emissions effectively during driving (2019) [6]. This method employed a rectangular area (the smallest bounding rectangle of the ellipse) to limit the search area, thereby improving the efficiency of the Dijkstra's algorithm. According to the time of one day, the Dijkstra's algorithm and an established database were used to identify the traffic

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

situation at the same time. The improved Dijkstra algorithm could not only reduce vehicle fuel consumption and emissions but also avoid time congestion. This method could be used for vehicle path planning based on dynamic traffic networks, reducing fuel consumption and emissions during driving, and improving urban environmental pollution.

Felipe Ribeiro Souza *et al.* applied the Dijkstra's method to tree diagram analysis, using mining blocks as nodes of the tree for analysis, and used to calculate the lowest cost route to transport mining blocks to their destination (2019) [7]. The transportation cost was reflected in the arc of the graph, and it could use Euclidean distance or transportation time to calculate the minimum path. The results obtained by the Dijkstra algorithm provided a non-operational path, to overcome this problem, adjustments were made through non-parametric equations. In this way, the transportation cost of each block of the model could be determined. Paths based on Euclidean distance and transit time tended to increase for deeper mines. Identifying the areas with the largest growth and quantifying their value correctly could improve the efficiency of mining planning.

Afonso Henriques Moreira Santos *et al.* used graph theory and the Dijkstra's shortest path algorithm for vertex location, completed tower positioning based on dynamic programming to find the optimal vertex set along the route (2019) [8]. This solution was used to solve the expansion planning problem of the new Transmission Line (TL), and its goal was to find a design solution with minimum cost. This method utilized the Dijkstra's shortest path algorithm to optimize the transmission line vertices and calculate the total cost from the source node to the sink node. The results showed that this method has a lower design cost than the original TL.

Jesús Balado *et al.* applied the Dijkstra pathfinding algorithm to the developed urban scene graph, and realized the task of directly using point clouds in the urban environment for pathfinding (2019) [9]. The method proposed in the paper could automatically set up a graph representing pedestrian navigable urban space, on which the safe and real routes of pedestrians under different movement skills could be calculated. The Dijkstra algorithm was utilized to develop safe routes in real-time graphics. The generated paths could be employed to make valid obstacle avoidance routes for pedestrians and wheelchairs.

Jinchuan Tang *et al.* studied the optimal path selection method based on the Dijkstra algorithm and combined with three probabilistic results for the design of Mission-Critical Push-To-Talk (MCPTT) system for 5G public safety disaster relief network (2019) [10]. The Dijkstra algorithm was used to select the best connection, delay, and trust routing. Aiming at the MCPTT system, the thesis proposed a routing method based on connection, delay, and trust to provide the best connection delay trust performance.

Based on the ArcGIS analysis tool, the Dijkstra algorithm was employed by Lingli Yu *et al.* for global path planning to accomplish the path planning and navigation control system design of 12 meters long driverless electric bus (2018) [11]. Based on the path planning and driving strategy, the optimal trajectory was generated by curve fitting technology, which fully considered the safety and dynamics of the driverless bus. This method could improve control accuracy, reduce the computational complexity, and promote driving efficiency.

Zheng Zhang *et al.* adopted an improved Dijkstra algorithm to determine the initial path of each task in the environment diagram describing the Automatic Guided Vehicle (AGV) in the grid method (2018) [12]. These authors proposed a collision-free routing method for AGVs in an automated warehouse based on collision classification. This method could deal with possible collisions in automated warehouses.

Feristah Dalkilic *et al.* used the Dijkstra's algorithm to reduce search space and runtime by applying stage-specific rules and utilized the algorithm in an intelligent itinerary planning system to assist passengers in itinerary planning (2017) [13]. The paper introduced a progressive path search algorithm to settle this problem, taking into account the number of transmissions and travel time. This method obtained a trip planning system by integrating route and timetable information from different transportation agencies. The system was managed to help users make better use of public transportation to simplify trip planning.

Sai Shao *et al.* applied a dynamic Dijkstra algorithm to determine the shortest path between any two adjacent nodes on the path (2017) [14]. The paper designed an electric vehicle routing scheme with variable charging time and travel time. Its purpose was to solve the dilemmas of electric vehicle mileage limitation and charging demand.

Georgios K.D. Saharidis *et al.* combined the Dijkstra's algorithm with a Mixed Integer Linear Programming (MILP) model to gain the optimal trip (2017) [15]. This multi-mode path solution could make people prefer to accept the minimum GreenHouse Gas (GHG) emission in various modes of transportation when traveling. This method could be used in the construction plan of the public transportation operation platform to achieve the best travel route for cutting emissions.

Tan Zhi *et al.* presented an improved ant colony algorithm to balance the energy consumption of wireless sensor networks by studying the theory of Dijkstra's algorithm (2015) [16]. The improved ant colony algorithm could increase the life cycle of wireless sensor networks.

W. C. Lu *et al.* used the Dijkstra's algorithm to settle the feasible air route planning issue, and the shortest path between the earned airport and the training area could minimize the impact on the crowd and the threat to the aircraft (2013) [17]. This paper aimed to explore the feasible air routes for light sports aircraft to minimize the impact on residential areas and the threat of terrain obstacles to the aircraft.

Deepak Gautam *et al.* applied the Dijkstra's algorithm to bypass obstacles and locate the shortest path from a given initial position to the final position (2013) [18]. The purpose of the paper to select the Dijkstra's algorithm was

to study the path planning of quadrotor helicopters in a closed known environment.

The earlier improved Dijkstra algorithm mainly focused on the improvement of the time complexity of the traditional Dijkstra algorithm and the promotion of the Dijkstra algorithm to different fields [19]–[26].

There are three main ways to improve the traditional Dijkstra algorithm. One is to analyze and improve the space complexity of the algorithm to improve storage efficiency and save space. The second is to analyze and improve the time complexity of the algorithm. The traditional Dijkstra algorithm has low efficiency and long running time. To improve operating efficiency and reduce time complexity, many documents have done a lot of work to improve this. The third aspect is to apply the algorithm to different fields, open up the application space of the algorithm and enrich the application field of the algorithm [5]–[26]. This article is to research in the third aspect, extending the Dijkstra algorithm to the optimal path on the curved surface to obtain a more accurate shortest path.

All the above studies are based on the Euclidean distance between neighbor nodes in the Dijkstra algorithm path planning [6]–[26]. The above papers rarely involve the optimal path of the surface [5]. Wolfgang Fink *et al.* utilized the Dijkstra algorithm to solve the 3D surface path planning task, they still used the Euclidean distance between the two nodes to calculate the optimal path, and their improvement was to take the weight as a composite factor, which comprehensively considered the effects of three-dimensional Euclidean distance, smoothness, roughness, height change, and other factors.

Many practical problems can be abstracted and transformed into the optimal surface path issue of mobile robots, such as the field rescue and material transportation route planning, the planetary ground exploration and development path planning, 3D game ground travel and war path planning, etc. The disadvantage of employing the traditional Dijkstra algorithm directly in surface path planning is that the calculation of each intermediate path weight is based on the Euclidean distance between adjacent nodes, which will bring about errors that cannot be ignored in the surface path planning. Because the optimal path obtained by calculating the Euclidean distance between nodes may not be on the surface, it is one of the most fundamental reasons for the optimal path error of the surface. All of these represent that new tools and methods are required to improve the surface optimal path planning process based on the traditional Dijkstra algorithm. The purpose of this paper is to establish a new general solution method with a higher precision, which is more suitable for the solution of the optimal path of the surface than the traditional Dijkstra model.

The main contributions of this paper are as follows: 1. Propose an approach for calculating the optimal path on a curved surface; 2. Improve the traditional Dijkstra algorithm for calculating the optimal path on a curved surface; 3. When compared with the traditional Dijkstra algorithm, the optimal path attained by the method proposed in this paper is more accurate, shorter, and smoother when calculating the optimal path of the surface. Especially for the case of a large number of nodes, a large per-unit scale, and a large surface ruggedness, the optimal path obtained by the method proposed in this paper has obvious advantages.

This article first presents the basic principle of the traditional Dijkstra algorithm (Section 2). Second, it explains the improved Dijkstra algorithm theory (Section 3). In this part, we utilize the Delaunay triangulation method to construct the surface map. The key to extending the classical Dijkstra algorithm is the invariance of the triangle side length when transforming a triangle on a surface into an equivalent triangle on a 2D plane. We convert the undevelopable surface channel into an equivalent two-dimensional passable channel, then solve the best path on the all two-dimensional passable channels, and finally obtain the best path of the surface through equivalent inverse transformation. We compare the simulation results of different surface examples to verify the effectiveness of the extended Dijkstra algorithm (Section 4). At the end of this paper, a conclusion is given (Section 5).

II. THE TRADITIONAL DIJKSTRA ALGORITHM

The initial work of the Dijkstra algorithm is only dealing with the shortest path between two points. Mathematically, these points must be represented by nodes in the graph network. Bellman Ford implemented the possibility of fixing a point and determining the shortest path to all other points in the graph. Paul carried out a common practical application, using this algorithm to figure out the shortest path between two cities, considering the street and highway to the destination. Sniedovich proposed a clear and structured step division to determine the minimum path between two points in a node network [1]–[4].

The Dijkstra algorithm (Algorithm 1) has a simple procedure. The essence of the traditional Dijkstra algorithm is to find out the shortest path between two nodes on a digraph $D = (N, W)$, where N is the set of all nodes and W is the set of weighted edges of connected nodes. The Dijkstra algorithm separates N into two sets, N_e and N_u. N_e is a set of all the end nodes of the determined shortest path to source node n_s. In the first step, N_e contains only n_s. N_u is the set of nodes to n_s with the undetermined shortest path. The nodes in N_u will be moved to N_e in ascending order of the shortest path length of the source node n_s until there are no nodes in the set N_u. The path that sequentially connects the source node n_s to all edges of any node n_{ti} is the shortest path from n_s to n_{ti}. The sum of the corresponding weights is the length of the Dijkstra algorithm's shortest path. So, the shortest path from the source node n_s to the target node n_t can be won.

From Algorithm 1, we can see that the logic of the traditional Dijkstra algorithm is to find the Euclidean distance between nodes, so it can be used to obtain the shortest path from the start point to the endpoint of a two-dimensional graph. However, when Algorithm 1 is directly applied to

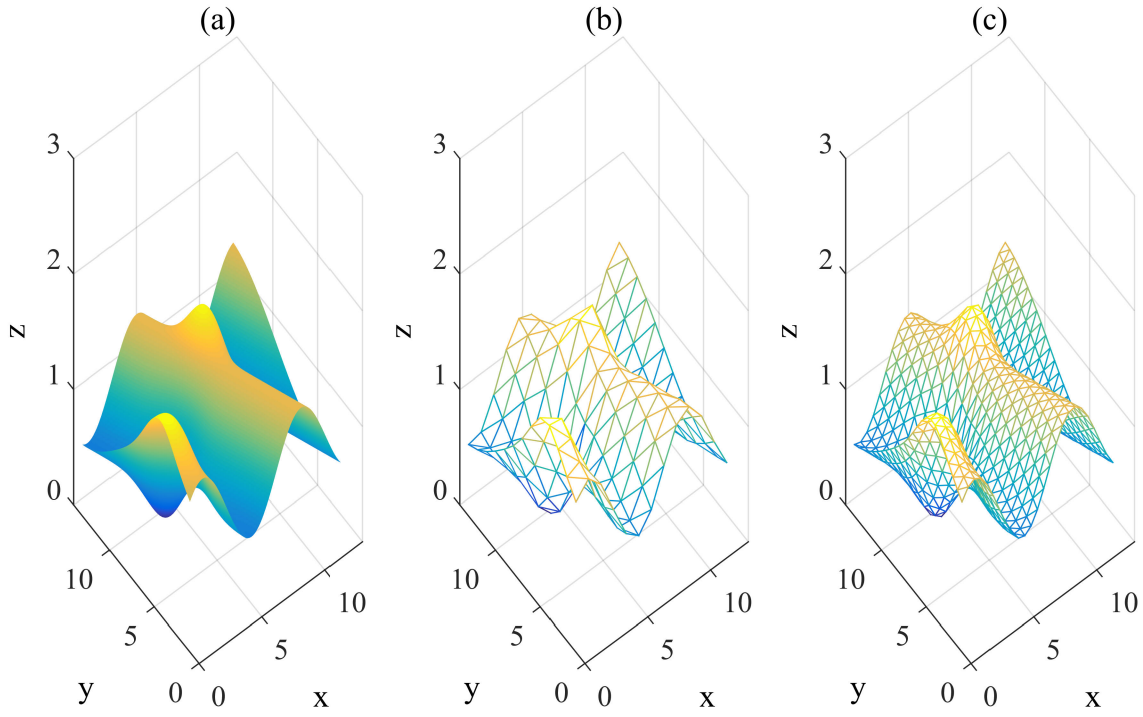


FIGURE 1. Surface Delaunay triangulation model. (a) The original surface. (b) 169 nodes Delaunay triangulation model. (c) 625 nodes Delaunay triangulation model.

calculate the shortest path between two points on a curved surface, the greater the curvature of the surface, the greater the error. The Euclidean distance between nodes and the distance between two points on the surface are two different issues.

III. THE EXTENDED DIJKSTRA ALGORITHM

In this section, the extended Dijkstra algorithm is introduced to resolve the surface optimal path planning task. The extended Dijkstra algorithm is an algorithm to transfer surface terrain map into a 2D map along the passable paths based on the invariance of the triangle side length for surface Delaunay Triangulation grid map.

The first step we shall do is to model a surface map by the Delaunay triangulation method. Delaunay triangulation is one of the most commonly used triangular mesh modeling methods. Compared with the square grid map, the Delaunay triangulation algorithm can provide more accurate surface information, and generate a smoother path [27], [28]. Compared with the digital point cloud method, it is simpler and more convenient.

The expression of the Delaunay triangle mesh subdivision algorithm makes each triangle unit has 12 adjacent units. Therefore, one non-boundary node can provide 12 feasible motion directions, thus this method can provide a smooth motion planning of the path. The Delaunay triangular grid map method [29], [30] is a map representation method using the triangular mesh as the cartographic unit, which can well express the fluctuation characteristics of the surface.

The accuracy and resolution of the surface map depend on the size of the triangle mesh. The smaller the mesh, the higher the accuracy. The choice of mesh size depends on the needing resolution requirement, as shown in Fig. 1.

For a particular deterministic surface map, choosing a larger mesh size map grid cell means fewer nodes and less resolution. Fig. 1 (a) shows an original surface map. And Fig. 1 (b) shows an example of the large mesh size surface map model of Fig. 1 (a), Fig. 1 (b) has 169 nodes to represent the surface characteristics information. Similarly, Fig. 1 (c) shows an example of the small mesh size surface map model of Fig. 1 (a), Fig. 1(c) has 625 nodes. Fig. 1 (c) has a higher resolution and can better represent the fluctuation characteristics of the surface of Fig. 1 (a). Fig. 1 (b) has fewer nodes, less mathematical complexity, and less program cost time. Therefore, in the actual design, the mesh size can be reasonably selected according to the required accuracy and time cost.

In Fig. 2 and Fig. 3, the red polygons are the obstructions that the robot cannot travel, and the other areas inside the map boundary are passable. The robot is R and the target is T. The numbers denoted in the Figs are the node serial numbers.

Fig. 2 (a) shows an original surface map example. The side length of each square (per-unit scale) is taken as 0.25 in Fig. 2. Using the traditional Dijkstra algorithm logic (Algorithm 1), we can get the black color curve optimal path $l_{ts(1,6,11,16)}$. Because the traditional Dijkstra algorithm is based on the Euclidean distance

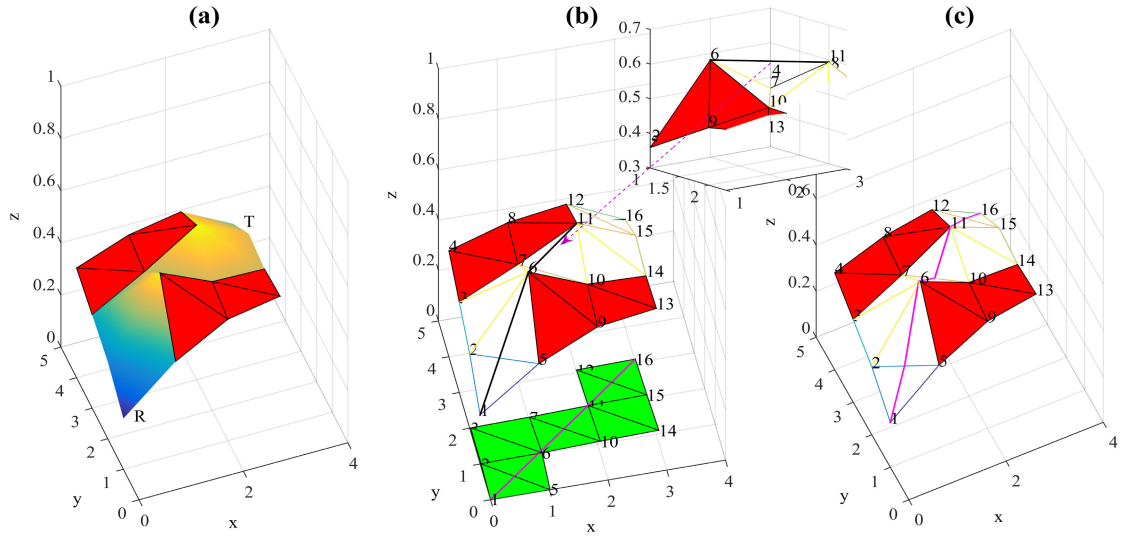


FIGURE 2. The extended Dijkstra algorithm theory diagram. (a) The original surface. (b) 16 nodes Dijkstra algorithm optimal path (black color) and the extended Dijkstra algorithm equivalent 2D passable path (green color). (c) 16 nodes the extended Dijkstra algorithm optimal path (magenta color).

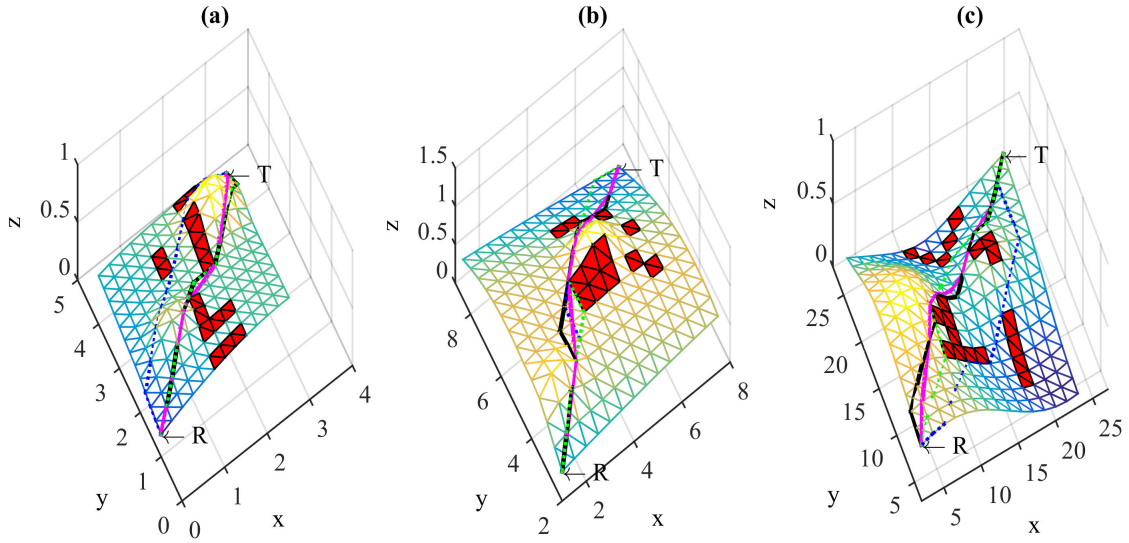


FIGURE 3. Four algorithms optimal path length comparison diagram. (The extended Dijkstra algorithm proposed in this paper is magenta solid line. The traditional Dijkstra algorithm is black solid line. The extended Dijkstra algorithm proposed in literature [5] is green dotted line. The A* algorithm is blue dotted line.) (a) 169 nodes example. (b) 196 nodes example. (c) 225 nodes example.

between adjacent nodes, the optimal path obtained is $l_{ts(1,6,11,16)} = \sqrt{(x_1 - x_6)^2 + (y_1 - y_6)^2 + (z_1 - z_6)^2} + \sqrt{(x_6 - x_{11})^2 + (y_6 - y_{11})^2 + (z_6 - z_{11})^2} + \sqrt{(x_{11} - x_{16})^2 + (y_{11} - y_{16})^2 + (z_{11} - z_{16})^2}$. Obviously, we can see from Fig. 2 (b), the black color curve optimal path $l_{ts(1,6,11,16)}$ maybe above, below, or on the surface. This causes errors in the optimal path of the surface.

Next, we showed the extended Dijkstra algorithm logic theory in Algorithm 2. The core idea of the extended Dijkstra algorithm is to find out the shortest path on a surface digraph $D_{sg} = (N_{sg}, W_{sg})$, where N_{sg} is the set of all nodes and W_{sg} is the set of weighted edges of connected nodes.

Then, we are going to introduce how to apply the invariant principle of triangular side length to convert a surface triangle into a triangle on the two-dimensional plane.

Shown in Fig. 2 (b), the surface nodes $1, 2, 5, 6$ are forming two three-dimensional triangulations $\Delta_{s(1,2,5)}$ and $\Delta_{s(2,5,6)}$ in white color. Keeping the invariance of the triangle side length, using Algorithm 2, the equivalent triangulation $\Delta_{p(1,2,5)}$ and $\Delta_{p(2,5,6)}$ in green color on the 2D plane can be

Algorithm 1 The Traditional Dijkstra Algorithm Framework

Input:

A bidirected graph with weights $D = (N, W)$, the set of all nodes N, the set of weighted edges of connected nodes W.

The source node n_s.

The target node n_t.

Output:

Shortest path and the length from the source node n_s to the target node n_t.

1: Initialization.
Visited nodes $N_e = n_s$.
$d(n_s) = 0$,
where $d(n_{ti})$ is the minimum cost of the shortest path from n_s to n_{ti}.
$N_u = N - N_e$,
where N_u is the set of nodes to n_s with the undetermined shortest path.
$d(n_{ti}) = min(W(n_s, n_{ti}))$, if n_{ti} is a successor to n_s, or else, $d(n_{ti}) = \infty$,
where $W(i, j)$ is the cost between node i and node j.

2: Repeat the following steps until there are no nodes in the set N_u.
2.1 Put the node in N_u that have the minimum cost to the old n_s as the new source n_s. Move the new source node n_s to N_e;
2.2 Set $d(n_{ti}) = d(n_s) + min(W(n_s, n_{ti}))$.

3: Find the shortest path from the source node n_s to the target node n_t.

4: **return** $d(n_t), N_e$.

found. In this way, the two-dimensional equivalent triangle adjacent to the two-dimensional equivalent triangle obtained in the previous step continues to travel along the passable passage from R to T until the last triangle with T node as the vertex of the triangle is obtained.

According to Algorithm 2, holding the length of each side of the triangle unchanged, the triangular meshes of the curved surface can be sequentially transformed into triangles on the two-dimensional plane one by one, to achieve an equivalent passable channel on the two-dimensional plane. Solve the best path of the two-dimensional equivalent passable triangular channel, and then traverse all the two-dimensional passable channels to attain the optimal path from the starting point to the endpoint of the equivalent two-dimensional plane. Finally, a simple inverse conversion of equivalent coordinates can secure the optimal path of the equivalent surface.

N_{dgpi} is the set of all 2D nodes and W_{dgpi} is the set of weighted edges of connected nodes on the 2D passable passage. Then, we separate N_{dgpi} into two sets, N_{edgpi} and N_{udgpi}. N_{edgpi} is a set of all the end nodes of the determined shortest path to the equivalent 2D source node n_{dsg}. In the first step, N_{edgpi} contains the only n_{dsg}. N_{udgpi} is the set of nodes to n_{dsg} with an undetermined shortest path. The nodes in N_{udgpi} will

Algorithm 2 The Extended Dijkstra Algorithm Framework

Input:

A surface graph $D_{sg} = (N_{sg}, W_{sg})$, the set of all nodes N_{sg}, the set of weighted edges of connected nodes W_{sg}. The source node n_{ssg}, R. The target node n_{stg}, T.

Output:

Shortest surface paths and their lengths from the source node n_{ssg} to the target node n_{stg}.

1: Convert the triangle which has the R robot's start point on the surface into an equivalent triangle on a 2D plane based on keeping the triangle side length unchanged. Continue this process to find a two-dimensional equivalent triangle adjacent to the two-dimensional equivalent triangle obtained in the previous step along a passable path from R to T until the last one tirangle with T node as the triangle vertex found.

2: Get the 2D equivalent passable passage graph $D_{dgpi} = (N_{dgpi}, W_{dgpi})$, where N_{dgpi} is the set of all nodes and W_{dgpi} is the set of weighted edges of connected nodes on the 2D equivalent passable passage graph.

3: Get the 2D source node n_{dsg}, and the 2D target node n_{dtg}.

4: Compute W_{dgpi} on the passable path in the two-dimensional plane.
4.1 Initialization.
Visited nodes $N_{edgpi} = n_{dsg}$.
$d(n_{dsg}) = 0$, where $d(n_{dtgpi})$ is the minimum cost of the shortest path from n_{dsg} to n_{dtgpi}.
$N_{udgpi} = N_{dgpi} - N_{edgpi}$, where N_{udgpi} is the the set of nodes to n_{dsg} with the undetermined shortest path.
$d(n_{dtgpi}) = min(W(n_{dgpi}, n_{dtgpi}))$, if n_{dtgpi} is a successor to n_{dsg}, or else, $d(n_{dtgpi}) = \infty$, where $W_{dgpi}(i, j)$ is the cost between the node i and j.
4.2 Repeat the following steps until there are no nodes in the set N_{udgpi}:
Set the node in N_{udgpi} that have the minimum cost to the old n_{dsg} as the new source n_{dsg}. Move the new source node n_{dsg} to N_{edgpi}. Set $d(n_{dtgpi}) = d(n_{dsg}) + min(W(n_{dgpi}, n_{dtgpi}))$.

5: Find the other passable passage from R to T in a different order, and repeat Step 1 - Step 4 until find the shortest paths for all passable passages.

6: Compute $d(n_{stg})$ and N_{esg} of the surface by equivalent inverse coordinate transformation from the shortest 2D path.

7: **return** $d(n_{stg}), N_{esg}$.

be moved to N_{edgpi} in ascending order of the shortest path length of the source node n_{dsg} until there are no nodes in the set N_{udgpi}. The path that sequentially connects the source node n_{dsg} to all edges of any node n_{dtgpi} is the shortest path from n_{dsg} to n_{dtgpi}. The sum of the corresponding weights is the length of the 2D passable passage's shortest path. So, the shortest path from the source node n_{dsg} to the target node n_{dtg} can be won.

Employing this method and by extension, we can easily take the 2D equivalent expansion channel of a surface passable path from the robot R's initial position to the target T's position. Thus, we can adapt the extended Dijkstra algorithm (Algorithm 2) to work out the optimal path on the acquired equivalent green 2D path below in Fig. 2 (b). A graph may have multiple passable channels. Computing all passable equivalent path on the 2D plane, we can get the shortest optimal path, the magenta color curve $l_{p(1,6,11,16)}$ shown in Fig. 2 (b). Finally, we use the equivalent coordinate transformation to get the surface optimal path from the robot R's initial position (n_{ssg}) to the target T's position (n_{stg}), the magenta color curve $l_{es(1,6,11,16)}$ shown in Fig. 2(c). And the magenta color curve is on the surface, not above or below the surface.

From Algorithm 2, we can see utilizing the extended Dijkstra algorithm, the surface smooth optimal path can be gotten. Therefore, in principle, the surface optimal path solved by this proposed method is more accurate and shorter than the surface optimal path solved by the traditional Dijkstra algorithm with the same nodes number.

Furthermore, the traditional Dijkstra algorithm optimal path is not a true path that falls on a triangulation mesh, and may often have path segments above or below the triangulation mesh. When the robot tracks along with the traditional Dijkstra algorithm optimal path, it cannot travel higher or lower than the curved surface and must travel against the curved surface (Refer to the detailed diagram in Fig. 2 (b), $l_{ts(6,11)}$ is the Euclidean space distance between node 6 and node 11, which is higher than the equivalent triangle blocks $\Delta_{s(6,7,10)}$ and $\Delta_{s(7,10,11)}$, not on these two triangles). Accordingly, the actual path length with the traditional Dijkstra algorithm optimal path will be much larger than the theory or simulation path length. The further the path is traveled or the greater the curvature of the curved surface, the greater the actual travel path is larger than the theoretical simulation path. And this path error is caused by a sharp bump or a large depression on the map, which may cause malfunctions such as the robot loses the direction of travel or walks on the wrong route. The improved Dijkstra algorithm optimal path is a true path that falls on an equivalent angulation mesh, so it is only slightly little different from the surface path of the real robot. On the same surface with the same number of nodes, the extended algorithm proposed in this paper can get a more accurate and smoother optimal path.

To further illustrate the characteristics of the extended algorithm in this paper, here is a comparative analysis of four algorithms, as shown in Fig. 3 and Table 1. The four algorithms for comparative analysis are the extended Dijkstra algorithm presented in this paper (magenta solid curve), the traditional Dijkstra algorithm (black solid curve), the extended algorithm introduced in literature [5] (green dotted curve), and the A* algorithm (blue dotted curve). For the extended algorithm proposed in [5], we comprehensively compared the simulation results of various factor parameter schemes proposed in the article [5], and finally selected the factor parameters $\alpha = 1/3$, $\beta = 1/3$, $\gamma = 1/3$ that can obtain a shorter path. For the A* algorithm, we compared the simulation results of a variety of heuristic functions $H(n)$ and selected the $H(n)$ that can obtain a shorter path, taking the Euclidean distance and the Manhattan distance into account.

TABLE 1. Four algorithms optimal path length comparison.

Figure number	Node number	Extended Dijkstra	Traditional Dijkstra	Extended Dijkstra [5]	A*
Fig. 3 (a)	169	4.509	4.645	4.770	4.645
Fig. 3 (b)	196	9.558	9.934	10.027	9.936
Fig. 3 (c)	225	30.828	32.342	33.222	32.345

It can be seen from Fig. 3 that only the optimal path obtained by the extended algorithm proposed in this paper is smooth and the optimal paths obtained by the other three algorithms are all non-smooth paths composed of polyline segments. The optimal path obtained by the algorithm proposed in this paper is based on a triangular mesh surface, while the optimal path obtained by the other three algorithms is the space Euclidean distance or Manhattan distance between the nodes. Therefore, the paths obtained by the other three algorithms all have line segments higher or lower than the triangle surface. On the one hand, it causes errors in path calculation, and on the other hand, it will cause problems such as malfunction or loss of direction when the robot is walking. More accurate paths have practical significance and value for occasions with higher precision requirements, such as surgical operations on fine surfaces or finishing on surfaces and other occasions with high error requirements.

From Fig. 3 and Table 1, the optimal paths acquired by the extended Dijkstra algorithm introduced in this paper are the shortest. As the number of nodes and the distance per-unit length increase, the path difference between the traditional Dijkstra algorithm and the algorithm proposed in this paper also increases, and the maximum error is about 4.91%. The maximum error of the path difference between the algorithm proposed in [5] and the algorithm in this paper is about 7.77%, and the maximum error of the path obtained by the A* algorithm and the algorithm in this paper is about 4.92%. The side length of each square (per-unit scale) is taken as 0.25 in Fig. 3 (a). In Fig. 3 (b), the side length of each square is 0.5, and in Fig.3 (c), it is 1.5.

When comparing these four algorithms in this paper, we can see that the purpose of the traditional Dijkstra algorithm is mainly to find the shortest path between nodes based on the Euclidean space distance. Literature [5] introduces the extended Dijkstra algorithm to select a flatter optimal path based on factors such as the shortest path, elevation difference, and ground smoothness. The A* algorithm is based on a reasonable selection of heuristic functions, and the search time for the shortest path is shorter than the traditional Dijkstra algorithm. And the extended algorithm proposed in this paper can get a smoother and shorter optimal path of

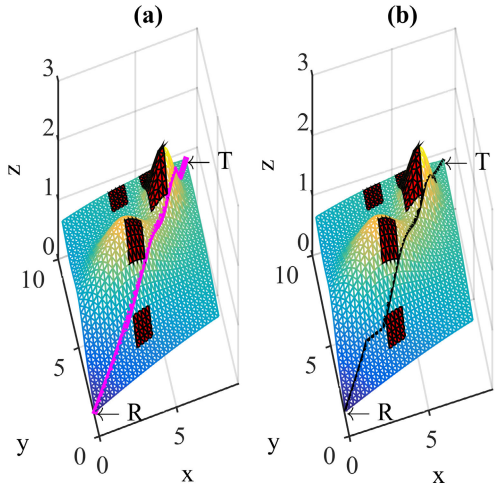


FIGURE 4. 1089 nodes example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

the surface than the shortest path obtained by the traditional Dijkstra algorithm. Both the curves in Figure 3 and the data in Table 1 confirm the effectiveness of the algorithm in this paper.

IV. MATLAB SIMULATION VERIFICATION

In this section, the MATLAB examples will further verify the effectiveness of this extended Dijkstra algorithm. First, 1-robot 1-target surface path planning simulation experiments with large numbers of nodes by the extended Dijkstra algorithm are conducted and compared the simulation results with the traditional Dijkstra algorithm simulation optimal path solutions. Furthermore, the simulation experiments of multi-robot and multi-target path planning are carried out. After many experiments, the results show that the improved Dijkstra algorithm introduced in this paper can provide smoother and shorter paths in 1-robot 1-target and multi-robot multi-target path planning tasks compared with the traditional Dijkstra algorithm.

In the Figs in this section, the red polygons are obstructions where the mobile robot cannot travel, and the other areas inside the map boundary are passable. The robot is R and the target is T in the Figs. The extended Dijkstra algorithm surface optimal path curve is in magenta, the black curve indicates the traditional Dijkstra algorithm surface optimal path.

A. 1-ROBOT 1-TARGET SURFACE PATH PLANNING

Firstly, we perform 1-robot 1-target surface path planning Matlab simulation experiments with large numbers of nodes. We use the Delaunay Triangulation Algorithm to build the surface environment with the required resolution in this section. And Algorithm 2's logic theory is employed to solve the surface optimal path.

In Fig. 4, we do a 1089 nodes surface simulation example experiment. The robot initial position is [0, 1, 0.120], and the target coordinate position is [8, 9, 1.190], as shown in Table 2.

TABLE 2. 1-robot 1-target surface optimal path comparison.

Figure number	Node number	Robot initial position	Target position	Extended Dijkstra	Traditional Dijkstra
Fig. 4	1089	[0, 1, 0.120]	[8, 9, 1.190]	11.812	12.582
Fig. 5	2401	[0, 1, 1.133]	[6, 7, 1.142]	8.798	10.818
Fig. 6	4225	[0, 1, 1.572]	[16, 17, 0.893]	23.082	24.875

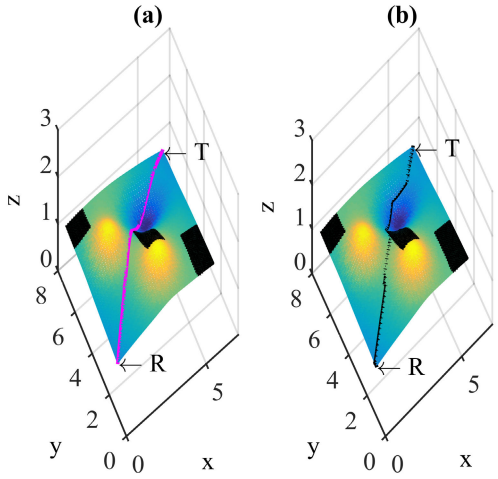


FIGURE 5. 2401 nodes example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

The magenta color curve in Fig. 4 (a) shows the optimal path calculated by the extended Dijkstra algorithm. It is smoother than the black curve shown in Fig. 4 (b) obtained by the traditional Dijkstra algorithm. And the optimal path length in magenta color is about 11.812, it is shorter than the black color optimal curve length 12.582.

We do a 2401 nodes surface simulation example experiment in Fig. 5. The robot's initial coordinate position in Fig. 5 is [0, 1, 1.133], the target position is [6, 7, 1.142], shown in Table 2. The magenta color optimal path length by the extended Dijkstra algorithm is about 8.798, and it is shorter than the black color optimal curve length 10.818 by the traditional Dijkstra algorithm, too.

A 4225 nodes surface path planning example is in Fig. 6. Table 2 exhibits the data of the simulation experiment. A conclusion drawn from the comparison of experimental data in Table 2 is that the extended Dijkstra algorithm is superior to the traditional Dijkstra algorithm in solving the 1-robot 1-target shortest path of the surface with large numbers of nodes, the extended Dijkstra algorithm optimal path is smoother and shorter. From Table 2, the maximum difference rate between the traditional Dijkstra algorithm and the extended Dijkstra algorithm is about 22.96%. The side length of each square (per-unit scale) is 0.25 in Fig. 4 and Fig. 6, and the per-unit scale is 0.125 in Fig. 5.

B. MULTI-ROBOT MULTI-TARGET SURFACE PATH PLANNING

Next, we perform multi-robot multi-target surface path planning Matlab simulation experiments. We extend

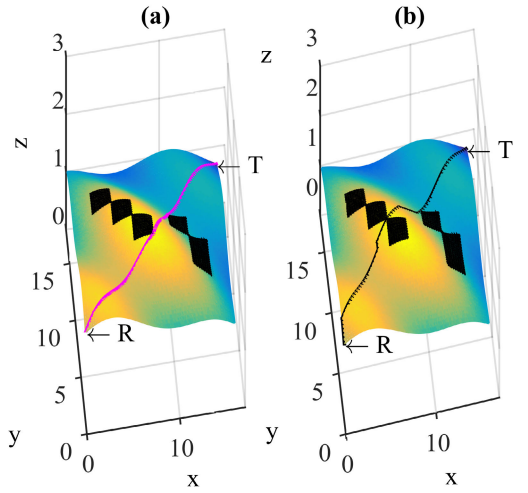


FIGURE 6. 4225 nodes example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

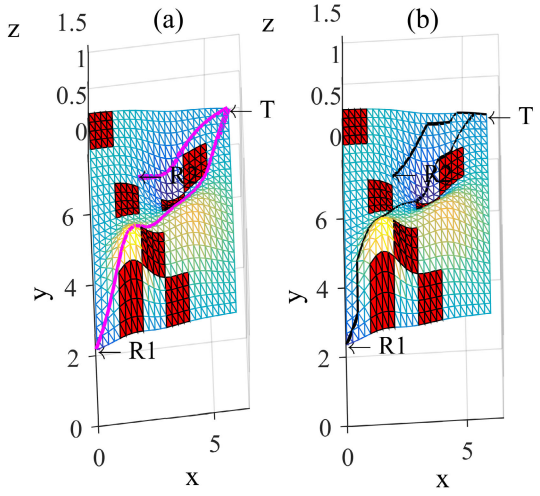


FIGURE 7. 625 nodes 2-robot 1-target example. (a) The extended dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

Algorithm 2 to multi-robot multi-target surface path planning tasks and carry out MATLAB simulation experiments in this section, keeping the simulation experiment settings consistent with the previous subsection.

In Fig. 7, we do a 625 nodes 2-robot 1-target surface path planning example experiment. The R1 robot's initial position is [0, 1, 0.569], the R2 robot's initial position is [2, 5.25, 0.789], and the target coordinate position is [6, 7, 0.742], as shown in Table 3. The magenta color curve length in Fig. 7 (a) from R1 to T is about 8.956, the magenta color curve length in Fig. 7 (a) from R2 to T is about 4.398. And, the black color curve length in Fig. 7 (b) from R1 to T is about 11.853, the black color curve length in Fig. 7 (b) from R2 to T is 4.732. So, according to the experimental curves and data in Figure 7 and Table 3, the optimal paths obtained by the extended Dijkstra algorithm are shorter than the optimal paths calculated by the traditional Dijkstra algorithm.

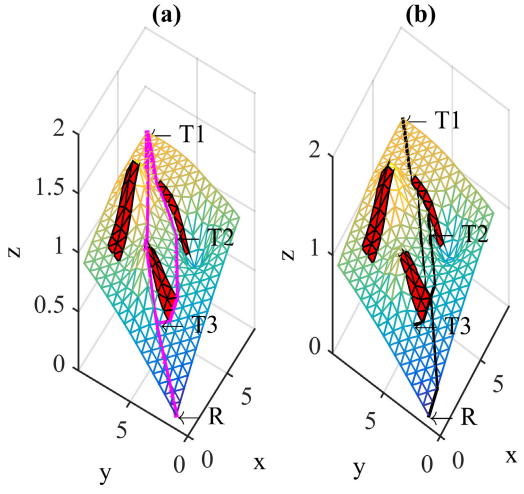


FIGURE 8. 289 nodes 1-robot 3-target example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

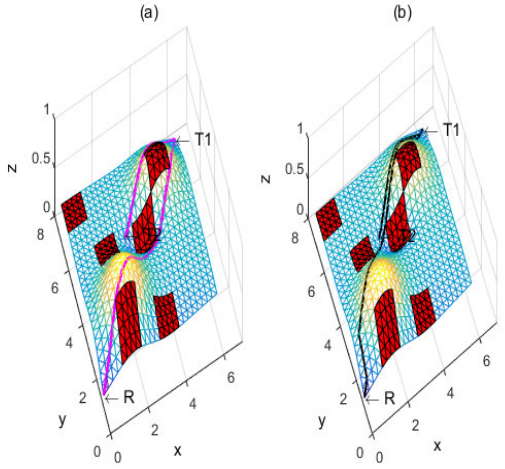


FIGURE 9. 625 nodes 1-robot 2-target example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

Fig. 8 shows a 289 nodes 1-robot 3-target surface path planning example. The R robot's initial position is [0, 1, 0.12], the T1 position is [8, 9, 1.190], the T2 coordinate position is [5, 4.5, 0.897], and the T3 position is [1.5, 3.5, 0.589], shown in Table 3. The magenta color curve length in Fig. 8 (a) from R to T1 is about 11.858, the magenta color curve length from T1 to T2 is about 5.615, and the magenta color curve length from T2 to T3 is about 4.224. The black color curve length in Fig. 8 (b) from R to T1 is about 14.626, the black color curve length from T1 to T2 is about 6.398, and the black color curve length from T2 to T3 is about 4.572. According to the experimental curves and data in Figure 8 and Table 3, the optimal paths obtained by the extended Dijkstra algorithm are always shorter and smoother than the optimal paths calculated by the traditional Dijkstra algorithm, too.

Fig. 9 shows a 625 nodes 1-robot 2-target surface path planning example. Fig. 10 is a 289 nodes 3-robot 2-target surface example. Fig. 11 is a 441 nodes 3-robot 2-target surface example. The experimental data are all filled in Table 3.

TABLE 3. Multi-robot multi-target surface optimal path comparison.

Figure number	Node number	Robot number &Target number	Robot initial position	Target position	Extended Dijkstra	Traditional Dijkstra
Fig. 7	625	$R=2, T=1$	$p_{r1}=[0,1,0.569]$ $p_{r2}=[2,5.25,0.789]$	$p_{t1}=[6,7,0.742]$	$op11=8.956, op21=4.398$	$op11=11.853$,$op21=4.732$
Fig. 8	289	$R=1, T=3$	$p_{r1}=[0,1,0.120]$	$p_{t1}=[8,9,1.190]$ $p_{t2}=[5,4.5,0.897]$ $p_{t3}=[1.5,3.5,0.589]$	$op11=11.858, op_{t1t2}=5.615$ $op_{t2t3}=4.224$	$op11=14.626, op_{t1t2}=6.398$ $op_{t2t3}=4.572$
Fig. 9	625	$R=1, T=2$	$p_{r1}=[0,1,0.118]$	$p_{t1}=[6,7,0.263]$ $p_{t2}=[2.5,4.75,0.335]$	$op11=8.952$,$op_{t1t2}=4.266$	$op11=11.256, op_{t1t2}=5.445$
Fig. 10	289	$R=3, T=2$	$p_{r1}=[0,1,0.120]$ $p_{r2}=[4,1,0.575]$ $p_{r3}=[2,5,1.073]$	$p_{t1}=[8,9,1.190]$ $p_{t2}=[7,1,0.860]$	$op11=11.858, op21=9.255$ $op32=7.500$	$op11=14.448, op21=11.554$ $op32=7.906$
Fig. 11	441	$R=3, T=2$	$p_{r1}=[0,1,0.092]$ $p_{r2}=[1,3,0.703]$ $p_{r3}=[2,1,0.269]$	$p_{t1}=[5,6,0.502]$ $p_{t2}=[1,5,0.579]$	$op11=7.536, op21=5.203$ $op31=6.024, op12=4.211$ $op22=2.010, op32=3.455$	$op11=8.512, op21=6.011$ $op31=7.723, op12=5.298$ $op22=2.010, op32=4.455$

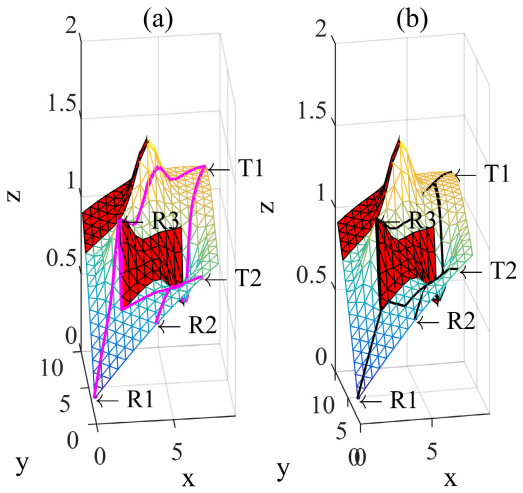


FIGURE 10. 289 nodes 3-robot 2-target example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

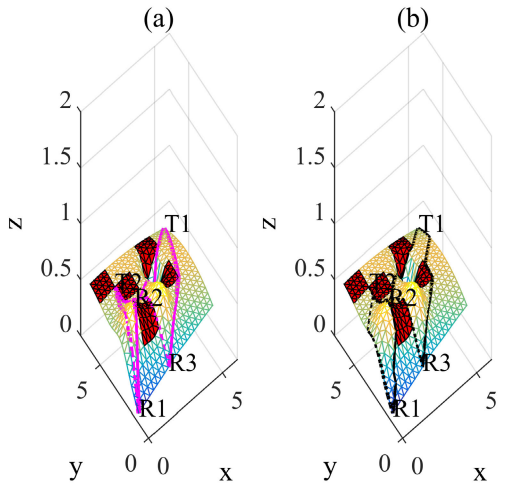


FIGURE 11. Comparion graph of 4225 nodes example. (a) The extended Dijkstra algorithm method. (b) The traditional Dijkstra algorithm method.

According to the experimental curves in these three figures and the simulation data in Table 3, it is easy to prove that the extended Dijkstra algorithm proposed by this subsection can obtain a more accurate and shorter surface optimal path than the traditional Dijkstra method. From Table 3, the maximum difference rate between the traditional Dijkstra algorithm and the extended Dijkstra algorithm is 32.35%. The per-unit scale is 0.25 in Fig. 7, Fig. 9, and Fig. 11, and in Fig. 8 and Fig. 10, the per-unit scale is 0.5.

The above simulation results show that the improved Dijkstra algorithm can still effectively calculate the optimal path of the surface in multi-robot and multi-target scenarios. Moreover, compared with the traditional algorithm, the curve is smoother and the path is always shorter.

The simulation research in this section finds that whether it is a single-robot single-object path planning task or a multi-robot multi-object path planning task, the extended algorithm can obtain a more accurate and smoother optimal path than the traditional algorithm, and usually, the path obtained by the extended algorithm is shorter.

V. CONCLUSION

To improve the error of the traditional Dijkstra algorithm when studying the surface optimal path task, we introduce the extended Dijkstra algorithm. The reason for the error of the traditional Dijkstra algorithm in researching the optimal path task of the surface is that it uses the Euclidean distance algorithm to calculate the path length between adjacent nodes. This is not a problem for the optimal path of a two-dimensional plane, but it will produce errors for the optimal path of a surface. Nowadays, more and more application

scenarios want to find the optimal path of the surface. Therefore, it is necessary to expand the research on the traditional optimal path algorithm to meet the needs of the new era.

In this paper, we first introduce an improved Dijkstra algorithm approach to solve the optimal path of the surface. We utilize the Delaunay triangulation method to model the surface environment. On the Delaunay triangulation map, we keep the triangular side length unchanged when transforming the triangular mesh on the surface into the corresponding triangle in the two-dimensional plane. Through the previous transformation, we attain the robot's accessible passages of the two-dimensional plane. Solve the shortest path in the two-dimensional plane, and then find the corresponding shortest path on the surface through equivalent coordinate inverse transformation.

Compared with the traditional Dijkstra algorithm, this approach extended the Dijkstra algorithm's research scope to the surface path planning field. The improved algorithm is suitable for the single-robot single-target project and the multi-robot multi-target project. And, compared with the traditional Dijkstra algorithm, the extended Dijkstra algorithm can acquire more accurate and shorter surface optimal path, which is demonstrated by the different MATLAB simulation examples in complex surface environments.

The smoother and shorter path obtained in this paper comes at the expense of increased time cost. In the future, we will combine the work of reducing time cost in the literature [31] to further study the method of reducing the time cost of this algorithm, so that the algorithm can be used in practical research more conveniently and quickly.

REFERENCES

[1] K. Wei, Y. Gao, W. Zhang, and S. Lin, "A modified Dijkstra's algorithm for solving the problem of finding the maximum load path," in *Proc. IEEE 2nd Int. Conf. Inf. Comput. Technol. (ICICT)*, Kahului, HI, USA, Mar. 2019, pp. 10–13, doi: 10.1109/INFOCT.2019.8711024.

[2] A. Alyasin, E. I. Abbas, and S. D. Hasan, "An efficient optimal path finding for mobile robot based on dijkstra method," in *Proc. 4th Scientific Int. Conf. Najaf (SICN)*, Al-Najef, Iraq, Apr. 2019, pp. 11–14, doi: 10.1109/SICN47020.2019.9019345.

[3] Yujin and G. Xiaoxue, "Optimal route planning of parking lot based on dijkstra algorithm," in *Proc. Int. Conf. Robots Intell. Syst. (ICRIS)*, Huai'an, China, Oct. 2017, pp. 221–224, doi: 10.1109/ICRIS.2017.62.

[4] M. A. Djojo and K. Karyono, "Computational load analysis of dijkstra, A, and floyd-warshall algorithms in mesh network," in *Proc. Int. Conf. Robot., Biomimetics, Intell. Comput. Syst.*, Jogjakarta, IN, USA, Nov. 2013, pp. 104–108, doi: 10.1109/ROBIONETICS.2013.6743587.

[5] W. Fink, V. R. Baker, A. J.-W. Brooks, M. Flammia, J. M. Dohm, and M. A. Tarbell, "Globally optimal rover traverse planning in 3D using Dijkstra's algorithm for multi-objective deployment scenarios," *Planet. Space Sci.*, vol. 179, pp. 1–9, Dec. 2019, doi: 10.1016/j.pss.2019.104707.

[6] D. Guo, J. Wang, J. B. Zhao, F. Sun, S. Gao, C. D. Li, M. H. Li, and C. C. Li, "A vehicle path planning method based on a dynamic traffic network that considers fuel consumption and emissions," *Sci. Total Environ.*, vol. 663, pp. 935–943, May 2019, doi: 10.1016/j.scitotenv.2019.01.222.

[7] F. R. Souza, T. R. Cámara, V. F. N. Torres, B. Nader, and R. Galery, "Mine fleet cost evaluation–Dijkstra's optimized path," *REM—Int. Eng. J.*, vol. 72, no. 2, pp. 321–328, Jun. 2019, doi: 10.1590/0370-44672018720124.

[8] A. H. M. Santos, R. M. D. Lima, C. R. S. Pereira, R. Osis, G. O. S. Medeiros, A. R. D. Queiroz, B. K. Flauzino, A. R. P. C. Cardoso, L. C. Junior, R. A. D. Santos, and E. L. C. Junior, "Optimizing routing and tower spotting of electricity transmission lines: An integration of geographical data and engineering aspects into decision-making," *Electr. Power Syst. Res.*, vol. 176, pp. 1–12, Jul. 2019, doi: 10.1016/j.epsr.2019.105953.

[9] J. Balado, L. Díaz-Vilariño, P. Arias, and H. Lorenzo, "Point clouds for direct pedestrian pathfinding in urban environments," *ISPRS J. Photogramm. Remote Sens.*, vol. 148, pp. 184–196, Feb. 2019, doi: 10.1016/j.isprsjprs.2019.01.004.

[10] J. Tang, G. Chen, X. Li, and J. P. Coon, "Route selection based on connectivity-delay-trust in public safety networks," *IEEE Syst. J.*, vol. 13, no. 2, pp. 1567–1576, Jun. 2019, doi: 10.1109/JSYST.2018.2813929.

[11] L. Yu, D. Kong, X. Shao, and X. Yan, "A path planning and navigation control system design for driverless electric bus," *IEEE Access*, vol. 6, pp. 53960–53975, 2018, doi: 10.1109/ACCESS.2018.2868339.

[12] Z. Zhang, Q. Guo, J. Chen, and P. Yuan, "Collision-free route planning for multiple AGVs in an automated warehouse based on collision classification," *IEEE Access*, vol. 6, pp. 26022–26035, 2018, doi: 10.1109/ACCESS.2018.2819199.

[13] F. Dalkäç, Y. Doäan, D. Birant, R. A. Kut, and R. Yälmaz, "A gradual approach for multimodel journey planning: A case study in izmir, turkey," *J. Adv. Transp.*, vol. 2017, pp. 1–14, 2017, doi: 10.1155/32017/5656323.

[14] S. Shao, W. Guan, B. Ran, Z. He, and J. Bi, "Electric vehicle routing problem with charging time and variable travel time," *Math. Problems Eng.*, vol. 2017, Jan. 2017, Art. no. 5098183, doi: 10.1155/2017/5098183.

[15] G. K. D. Saharidis, D. Rizopoulos, A. Fragkogios, and C. Chatzigeorgiou, "A hybrid approach to the problem of journey planning with the use of mathematical programming and modern techniques," *Transp. Res. Procedia*, vol. 24, pp. 401–409, Oct. 2017, doi: 10.1016/j.trpro.2017.05.094.

[16] T. Zhi and Z. Hui, "An improved ant colony routing algorithm for WSNs," *J. Sensors*, vol. 2015, pp. 1–4, Dec. 2015, doi: 10.1155/2015/438290.

[17] W. C. Lu, M. T. Lee, and M. W. Wang, "Route planning for light-sport aircraft in constrained airspace," *Procedia Eng.*, vol. 67, pp. 140–146, 2013, doi: 10.1016/j.proeng.2013.12.013.

[18] D. Gautam and C. Ha, "Control of a quadrotor using a smart self-tuning fuzzy PID controller," *Int. J. Adv. Robotic Syst.*, vol. 10, pp. 1–9, Aug. 2013, doi: 10.5772/56911.

[19] T.-H. Kim and I.-C. Park, "High-throughput and area-efficient MIMO symbol detection based on modified Dijkstra's search," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 7, pp. 1756–1766, Jul. 2010, doi: 10.1109/TCSI.2009.2034235.

[20] Z. Pan, L. Yan, A. C. Winstanley, A. S. Fotheringham, and J. Zheng, "A 2-D ESPO algorithm and its application in pedestrian path planning considering human behavior," in *Proc. 3rd Int. Conf. Multimedia Ubiquitous Eng.*, Qingdao, China, Jun. 2009, pp. 485–491, doi: 10.1109/MUE.2009.86.

[21] A. Chen, A. K.-S. Wong, and C.-T. Lea, "Routing and time-slot assignment in optical TDM networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1648–1657, Nov. 2004, doi: 10.1109/JSAC.2004.833832.

[22] Bast, Mehlhorn, Schäfer, and Tamaki, "A heuristic for Dijkstra's algorithm with many targets and its use in weighted matching algorithms," *Algorithmica*, vol. 36, no. 1, pp. 75–88, May 2003, doi: 10.1007/s00453-002-1008-z.

[23] M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Nashville, TN, USA, Oct. 2000, pp. 2316–2320, doi: 10.1109/ICSMC.2000.886462.

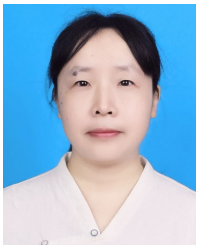
[24] D. Cavendish and M. Gerla, "On routing with QOS constraints in ATM networks," in *The International Federation for Information Processing*, A. Tantawy, Ed. Boston, MA, USA: Springer, 1994.

[25] R. V. Helgason, J. L. Kennington, and B. D. Stewart, "The one-to-one shortest-path problem: An empirical analysis with the two-tree dijkstra algorithm," *Comput. Optim. Appl.*, vol. 2, no. 1, pp. 47–75, Jun. 1993, doi: 10.1007/BF01299142.

[26] R. B. K. Dewar, S. M. Merritt, and M. Sharir, "Some modified algorithms for Dijkstra's longest upsequence problem," *Acta Inf.*, vol. 18, no. 1, pp. 1–15, Feb. 1982, doi: 10.1007/BF00625277.

[27] M. Liu, "Robotic online path planning on point cloud," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1217–1228, May 2016, doi: 10.1109/TCYB.2015.2430526.

[28] A. Stumpf, S. Kohlbrecher, D. C. Conner, and O. von Stryk, "Supervised footstep planning for humanoid robots in rough terrain tasks using a black box walking controller," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Madrid, India, Nov. 2014, pp. 287–294, doi: 10.1109/HUMANOIDS.2014.7041374.
[29] C.-C. Sun, G. E. Jan, S.-W. Leu, K.-C. Yang, and Y.-C. Chen, "Near-Shortest path planning on a quadratic surface with $O(n \log n)$ time," *IEEE Sensors J.*, vol. 15, no. 11, pp. 6079–6080, Nov. 2015, doi: 10.1109/JSEN.2015.2464271.
[30] Z. Shiller and J. C. Chen, "Optimal motion planning of autonomous vehicles in three dimensional Terrains," in *Proc. Int. Conf. Robot. Autom.*, Cincinnati, OH, USA, 1990, pp. 198–203, doi: 10.1109/ROBOT.1990.125972.
[31] M. Luo, X. Hou, and S. X. Yang, "A multi-scale map method based on bioinspired neural network algorithm for robot path planning," *IEEE Access*, vol. 7, pp. 142682–142691, 2019, doi: 10.1109/ACCESS.2019.2943009.

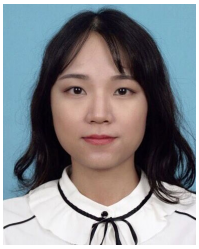


MIN LUO was born in Sichuan, China, in 1978. She received the B.S. degree in automation from Chongqing University, in 2001, and the M.S. degree in electrical engineering and information from Southwest Petroleum University, in 2009. She is currently pursuing the Ph.D. degree in automation engineering with the University of Electronic Science and Technology of China, Sichuan.

Since 2001, she has been a Lecturer with the School of Electrical Engineering and Information, Southwest Petroleum University. Her research interests include robot path planning and control theory and application.

XIAORONG HOU was born in Shanxi, China, in 1966.

He is currently a Professor with the School of Automation Engineering, University of Electronic Science and Technology of China. He has published over 90 research articles and two monographs. His research interests include control theory, intelligent systems, symbolic computation, and real algebraic geometry.



JING YANG was born in Sichuan, China, in 1989.

She is currently a Postdoctoral Researcher with the School of Automation Engineering, University of Electronic Science and Technology of China. Her research interests include nonlinear control theory and fractional-order systems.