Ocean Engineering 266 (2022) 112873

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng

Path planning of unmanned surface vessel in an unknown environment based on improved D*Lite algorithm

Jiabin Yu [a,b,c], Meng Yang [a], Zhiyao Zhao [a,b,c], Xiaoyi Wang [a,b,c], Yuting Bai [a,b,c], Jiguang Wu [a], Jiping Xu [a,b,c,*]

[a] School of Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China
[b] China and Beijing Laboratory for Intelligent Environmental Protection, Beijing Technology and Business University, Beijing 100048, China
[c] Key Laboratory of Industrial Internet and Big Data, China National Light Industry, Beijing Technology and Business University, Beijing 100048, China

ARTICLE INFO

Keywords:
Smooth path planning
D*Lite algorithm
Dubins curve
Unknown obstacle environment
Unmanned surface vessel

ABSTRACT

To solve the problem of path planning for an unmanned surface vessel in an unknown environment, this paper proposes a path planning algorithm based on the improved D*Lite algorithm. First, to improve the computational efficiency of the traditional D*Lite algorithm, the path cost function is improved to reduce the expansion range of nodes, and the node expansion direction is limited to avoid double node computation. Second, the path planned by the traditional D*Lite algorithm on the grid map is suboptimal; so, the inverse distance weighted (IDW) interpolation method is applied to shorten the path length. To improve the smoothness of the planned path by the traditional D*Lite algorithm, the Dubins algorithm is introduced for local path smoothing, and a smooth collision-free path that conforms to the motion dynamics of the unmanned surface vessel is obtained. The proposed algorithm was verified by experiments in known and unknown environments. The performance of the proposed algorithm was compared with the traditional D*Lite algorithm in terms of planning time, path length, and path smoothness. The results revealed that the proposed algorithm has the shortest path length, a shorter planning time, and the best smooth path.

1. Introduction

As a recent innovation, unmanned surface vessels (USVs) are widely used in the field of water quality monitoring and sampling. They are used to replace traditional fixed monitoring and manual sampling so as to achieve efficient and flexible water sample collection and water environment monitoring. The path planning technology is an important part of the USV navigation system. It can plan a collision-free path suitable for the motion dynamics of the USV so that the USV can quickly reach the water quality sampling point. It is the key technology of the USV automatic driving. The path planning technology of USV usually refers to the path planning research on a ground robot or unmanned aerial vehicle (UAV), but their motion dynamics are not the same as that of the USV; so, their path planning algorithms are not applicable to USVs. Moreover, there are usually unknown obstacles in the water environment, which will affect the normal operation of USVs. Therefore, it is essential to research the path planning of USVs in unknown environments.

Many studies on the path planning problem have been conducted for decades, and the path planning methods can be divided into two main categories: global path planning, such as the A*algorithm (Hart et al., 1968), breadth-first search (BFS) (Nascimento et al., 2018), depth-first search (DFS) (Jose and Antony, 2016), Dijkstra algorithm (Dijkstra et al., 1959), rapidly exploring random tree (RRT) (LaValle et al., 1998) and probabilistic roadmap (PRM) (Kavraki et al., 1996). Local path planning includes the artificial potential field (APF) (Lin et al., 2019), dynamic window approach (DWA) (Saranrittichai et al., 2013) and time elastic band (TEB) (Rösmann et al., 2012). Global path planning is an offline type of planning, which cannot be applied to unknown environments. The local path planning ignores global factors and easily falls into local optimum. With the development of computer technology, many bionic intelligence algorithms have been applied to the field of path planning, such as the genetic algorithm (GA) (Bashiri et al., 2018), ant colony optimization (ACO) (Cil et al., 2020), particle swarm optimization (PSO) (Dabiri et al., 2017), cuckoo search (CS) (Mohanty and Parhi, 2016) and neural network (NN) (Yoo, 2019). These algorithms require constant tuning of parameters and multiple iterations to achieve optimization, which is a huge workload. Therefore, many hybrid algorithms have been proposed to avoid the disadvantages of

* Correspondence to: School of Artificial Intelligence Beijing Technology and Business University, No.11/33 FuCheng Road, HaiDian District Beijing, 100048, P.R. China.
E-mail address: xujp@th.btbu.edu.cn (J. Xu).

https://doi.org/10.1016/j.oceaneng.2022.112873
Received 20 May 2022; Received in revised form 10 September 2022; Accepted 8 October 2022
Available online 3 November 2022

a single algorithm while inheriting the advantages of each at the same time. Zhu et al. (2019) proposed a novel method based on the fusion of the A* algorithm and DWA; the DWA based on the evaluation function was applied to implement dynamic path planning to guarantee the ability of obstacle avoidance while ensuring the global optimality of the path. Wang et al. (2019) proposed a hybrid genetic–cuckoo intelligence algorithm. The GA is used as the global search algorithm and is combined with the cuckoo algorithm to improve the local search ability of the algorithm, which can achieve reasonable obstacle avoidance in three-dimensional space under multiple constraints. Abubakr et al. (2018) proposed a reduced fuzzy logic controller, which optimizes the weights of the DWA in real time according to the distribution of obstacles around the robot. It can better realize the path planning and obstacle avoidance processing in a complex environment. Most of these algorithms focus on path planning in known environments, but there are usually many unknown obstacles in the actual water environment. Therefore, path planning in an unknown environment is gradually becoming the focus of current research.

In this regard, Feng et al. (2020) proposed a self-adaptive dynamic window algorithm (SDWA) to dynamically search both global and local optimal paths; the weight parameter of the objective function is adaptively controlled based on the distance between the multi-robot system and obstacles, which aims to balance the speed and safety of the multi-robot system. Yaoming et al. (2021) proposed a bio-inspired path planning algorithm in three-dimensional space. The algorithm imitates the basic mechanisms of plant growth, including phototropism, negative geotropism, and branching, has the advantages of a fast path planning speed and fewer route points, and can achieve the effect of low delay real-time path planning. Maurović et al. (2017) proposed a path planning algorithm for active simultaneous localization and mapping (SLAM) that continuously improves a robot's localization while moving smoothly. The algorithm is based on the D* shortest path graph search algorithm with negative edge weights for finding the shortest path, taking into account localization uncertainty. Qi et al. (2020) proposed an algorithm termed multi-objective dynamic rapidly exploring random (MOD-RRT*). The algorithm generates a state tree structure as prior knowledge during initial planning, which can choose the best node f among several candidates within a short time when the current path is not feasible. Moreover, the D*Lite algorithm is an efficient dynamic path planning method that can achieve real-time obstacle avoidance and fast path replanning in unknown environments, and the algorithm has been widely used for path planning in unknown environments. Osmankovic et al. (2017) proposed a multi-stage technique based on the fast D*Lite algorithm for global path cost computation while employing the model predictive control (MPC) planning paradigm for solving the constrained optimal control problem for the purpose of local planning. Huang and Zhou (2020) proposed an improved D*Lite algorithm. This algorithm introduced the concept of a combination of lazy line-of-sight and distance transformation so that the re-planned path can avoid sudden obstacles. Sebastian and Ben-Tzvi (2019) described a novel physics-based path planning architecture for autonomous navigation of tracked vehicles under rough and unknown terrain conditions. The architecture uses the D*Lite algorithm working on a 2D grid representation of the terrain as the high-level planner, which is more effective in providing an optimal feasible path. Although the D*Lite algorithm performs well in unknown environments, the traditional D*Lite algorithm still has some shortcomings, which can be summarized as follows.

First, the traditional D*Lite algorithm has the drawbacks of slow planning efficiency, a large expansion range, and double computation of nodes. To solve these problems, Le et al. (2017) proposed the new algorithm D*Lite with Reset (D*LR), which sets a threshold on the ratio of the traversed path length to the total path length. If the threshold is exceeded, resetting of the D*Lite algorithm will yield a better performance in a complex environment. Przybylski and Putz (2017) proposed a new D*Extra Lite algorithm, with re-initialization of the affected search-space achieved by search-tree branch cutting without the use of complex operations on the open list. It has higher computational efficiency than the traditional D*Lite algorithm in unknown environments. Reyes et al. (2018) proposed an extension to the D*Lite algorithm, called D*Lite with Poisoned Reverse, which can detect whether the node update process is being double computed, remove the affected nodes, reduce node expansion, and avoid path backtracking in unknown environments.

Second, the D*Lite algorithm is mainly used for grid maps generated based on the grid method, where there are restrictions on the direction of movement and the final path length is usually suboptimal. To optimize the path length, Chao et al. (2019) proposed a sampling-based algorithm, which combines the principle of random tree star (RRT*) and D*Lite, and used the expansion strength of the grid search strategy from D*Lite to quickly find a high-quality initial path to accelerate the convergence rate in RRT*. It refines the length of existing paths continually by sampling the search-graph obtained from the grid search process. Ferguson and Stentz (2005) proposed a Field*D algorithm that uses linear interpolation to compute the exact path cost at any location within each grid, capable of generating a path of the shortest length in both uniform and non-uniform path cost environments.

The planned paths by the traditional D*Lite algorithm on the grid map usually have too many turning points. Such unsmoothed paths are not suitable for the motion dynamics of the USV, and they cannot be practically applied. Therefore, how to smooth the paths in unknown environments is also a research focus of path planning. To solve this problem, Ali et al. (2020) introduced the Markov decision process trajectory evaluation model in the iterative process of the ACO algorithm and proposed a novel reward policy to filter and reduce the sharpness in the global path generated in the grid environment to generate a smooth path. Yuan et al. (2019) proposed an NN algorithm using a four-layer network structure, and the energy function was designed as an evaluation function of the network. The smart cart can adjust the cart to plan a smooth path based on the trend of the set path points. Pattnaik et al. (2022) proposed a hybrid particle swarm and chemical reaction optimization (HPCRO) algorithm. A new objective function was formulated by considering the steering angle and the obstacles and goal positions of the mobile robot to find the feasible, shortest, and smoothest path. Cui et al. (2018) designed a heuristic strategy based on the elastic band theory and adopted Dubins curves as the basic path segments; this has been employed to determine the nodes and branches for obstacle avoidance. Their approach can handle the obstacle avoidance problem and planning path for the robots with pose constraints. Ghadiry et al. (2020) developed a pulley algorithm (PA) to convert the ETSP optimal solution to a kinematically feasible optimal Dubins path that can be tracked by Dubins' vehicles. Under this algorithm, the robots can move continuously, as close as possible to the optimal ETSP path, without having to stop and turn on the spot at the turning points of the path. Vautier et al. (2019) developed a geometrical construction of the shortest Dubins path in a discontinuous orientation-restricted environment. With the use of bitangents of circles, the shortest path was built from one pose to another by adding a middle turn, making a smooth path.

In summary, the traditional D*Lite algorithm is still insufficient in unknown environments. Moreover, the path planning problem becomes increasingly complex, and the requirements for the algorithm are higher. First, the computational efficiency of the algorithm determines the effectiveness of the algorithm in an unknown environment. When encountering unknown obstacles, if the USV cannot re-plan a new path on time, this will affect the safety. Second, the grid map restricts the moving direction of the USV, and the planned path is not optimal in length. In addition, the traditional D*Lite algorithm has too many turning points in the planned path in a complex environment, which is not feasible for the USV. To solve these problems, an improved D*Lite algorithm for path planning of USVs in an unknown environment is proposed in this paper. The main contributions of this work are as follows:

(1) The computational efficiency of the D*Lite algorithm in an unknown environment is improved by improving the path cost function to reduce the expansion range of nodes and by limiting the direction of node expansion to avoid double node computation.
(2) To optimize the path length planned by the traditional D*Lite algorithm, the inverse distance weighted (IDW) interpolation method is used to increase the selectable directions and shorten the path length.
(3) Dubins curve is introduced for local path smoothing to solve the problem that the D*Lite algorithm cannot smooth the global path. The smoothness of the planned path on the grid map is improved, and a smooth collision-free path that conforms to the motion dynamics of the USV is obtained.

The rest of this paper is organized as follows. Section 2 provides the preliminaries and problem formulation. Section 3 presents the smooth path planning algorithm based on the improved D*Lite algorithm. Section 4 presents and discusses the simulation experimental results. Finally, Section 5 concludes the paper.

2. Preliminaries and problem formulation

2.1. D*Lite algorithm

The D*Lite algorithm (Koenig and Likhachev, 2005) is a dynamic path planning algorithm proposed by Sven Koenig and Maxim Likhachev based on the LPA* (Koenig et al., 2004) algorithm. It has the advantage that it can improve the efficiency of path replanning by using the node information of the initially planned path; so, it can be applied to dynamic path finding of robots in unknown environments. Unlike the LPA* algorithm that explores the path from the starting point to the target point, the D*Lite algorithm uses the idea of the D* algorithm (Stentz, 1993) reverse search, exploring from the target point to the starting point. The core of the D*Lite algorithm assumes that the unknown regions are all free spaces and implements path planning incrementally. The D*Lite algorithm finds the minimum distance from the target point to nearby nodes by minimizing $rhs(s)$, and $rhs(s)$ denotes the path cost from the current node to the target node based on the dynamic SWSF-FP algorithm (Ramalingam and Reps, 1996) where it is the value of the right-hand side (rhs) of the grammar rule. The $rhs(s)$ is obtained according to the following equation:

$$rhs(s) = \begin{cases} 0, & \text{if } s = s_{goal} \\ \min\limits_{s' \in Succ(s)} (c(s,s') + g(s')) & \text{otherwise} \end{cases} \tag{1}$$

where node s' is a node adjacent to node s, Assume that S denotes the finite set of nodes in a graph, and $Succ(s) \subset S$ denotes the set of successors of a node s, $s \in S$, which indicates all other nodes that can be moved from this node s. $c(s,s')$ denotes the cost of moving from node s to $s' \in Succ(s)$. The $g(s)$ terms directly correspond to the g-values of the A* algorithm, which is the actual path cost from the current expansion node s' to node s_{goal}, and the rhs-values are one-step lookahead values based on the g-values and are thus potentially better informed than the g-values.

To find the optimal path, the current node s calculates the evaluation function $k(s)$ of the surrounding successor nodes and compares the size of their $k(s)$. The D*Lite algorithm continuously updates a priority queue U, which is used to store the nodes to be updated in order of the size of the evaluation function $k(s)$. The smallest $k(s)$ of these successor nodes s' is the expansion node. Then, the node s' with the smallest $k(s)$ as the current node is used to calculate the k-value of its successor nodes. The $k(s)$ contains two components $k(s) = [k_1(s); k_2(s)]$ as follows:

$$\begin{cases} k_1(s) = \min(g(s), rhs(s)) + h(s_{start}, s) + k_m \\ k_2(s) = \min(g(s), rhs(s)) \end{cases} \tag{2}$$

$$h(s_{start}, s) = \begin{cases} 0, & \text{if } s = s_{start} \\ c(s,s') + h(s', s_{start}) & \text{otherwise} \end{cases} \tag{3}$$

where $h(s_{start}, s)$ is the heuristic function that represents the path cost from the start node s_{start} to node s. The evaluation function $k(s)$ determines the order of node expansion, and the node with the smallest $k(s)$ among the adjacent nodes is used as the next expansion node. The size of $k_1(s)$ is compared in priority, and the size of $k_2(s)$ is compared if $k_1(s)$ is the same. Since each movement of the USV takes the current node as the new starting point s_{start}, updating the $h(s_{start}, s)$ value of each node causes the valuation function to be recalculated. If the number of nodes is large, this will greatly reduce the computational efficiency of the algorithm. Therefore, k_m is introduced to modify $k(s)$ in the calculation as a key modifier. Before the USV moves, $k_m = 0$. For each step the USV moves, $k_m := k_m + h(s_{start}, s'_{start})$ and the current position become the new starting point s'_{start}. The $h(s_{start}, s'_{start})$ term represents the path cost of each movement of the USV. This improves the overall efficiency of computing the node evaluation function, so that the order of nodes stored in the queue to be detected does not change and the queue does not need to be reordered.

2.2. Dubins curves

Under the condition that the direction of the starting and target points are specified, Dubins curves (Dubins, 1957) are the shortest paths connecting two two-dimensional planes when the curvature constraint is satisfied. The shortest path consists of exactly three path segments and presents a sequence CCC or CSC,where C (for "circle") is an arc, and S (for "straight") is a line segment. Each arc C has two options — turning left (L) or turning right (R). The Dubins curve set D includes six admissible paths, $D = \{LSL, LSR, RSL, RSR, RLR, LRL\}$. Assuming that r_{c1} and r_{c2} are the turning radii of the USV, θ_1 and θ_2 are the respective directional angles of the starting point and the ending point at the path turning points, and d is the distance between the starting point and the ending point. If d satisfies Eq. (4), The length of the CCC curve is always longer than that of the CSC curve, and the shortest Dubins curve is necessarily one of the CSC curve; so, only the CSC curve is considered instead of the turning point in this paper.

$$d > \sqrt{4r_{c1}r_{c2} - (|r_{c1}\cos\theta_1| + |r_{c2}\cos\theta_2|) \\ + |r_{c1}\sin\theta_1| + |r_{c2}\sin\theta_2|} \tag{4}$$

The key to calculate the Dubins curve is to find the centers of the starting and ending circles, and the tangent points of the line segment to the two circles. As shown in Fig. 1, assuming that $s_{start}(x_1, y_1)$ and $s_{end}(x_2, y_2)$ are the starting and ending points of the segment respectively, and the following parameters can be derived.

The centers of the starting circle $s_{c1}(x_{c1}, y_{c1})$ and ending circle $s_{c2}(x_{c2}, y_{c2})$ are given by

$$\begin{cases} s_{c1} = (x_1 + r_{c1}\cos(\theta_1 \pm \pi/2), y_1 + r_{c1}\sin(\theta_1 \pm \pi/2)) \\ s_{c2} = (x_2 + r_{c2}\cos(\theta_2 \pm \pi/2), y_2 + r_{c2}\sin(\theta_2 \pm \pi/2)) \end{cases} \tag{5}$$

when the arc is turning left, the angle is $\theta_1 + \pi/2$; when the arc is turning right, the angle is $\theta_1 - \pi/2$.

The angle α between the line connecting the center of the circles l_c and the common tangent l_s is given by

$$\alpha = \begin{cases} \arcsin((r_{c2} - r_{c1})/l_c), \text{external common tangent} \\ \arcsin((r_{c2} + r_{c1})/l_c), \text{internal common tangent} \end{cases} \tag{6}$$

The angle β between the line connecting the center of the circles and the positive direction of the x-axis is given by

$$\beta = \arctan\left(\frac{y_{c2} - y_{c1}}{x_{c2} - x_{c1}}\right) \tag{7}$$

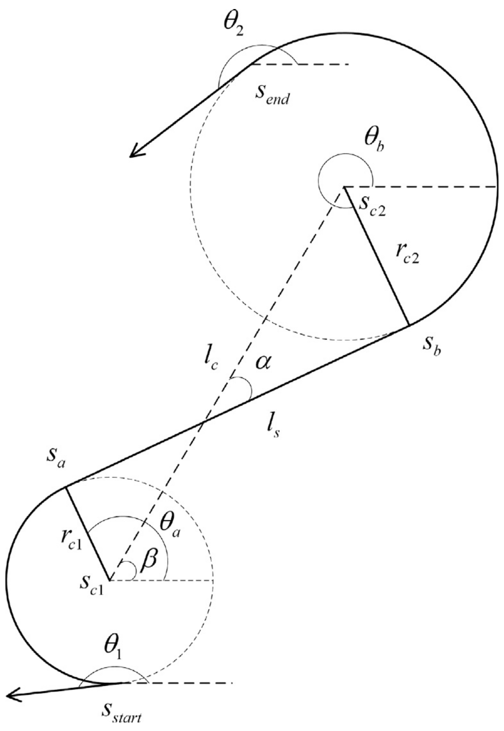

Fig. 1. Calculating the Dubins curve.

Table 1
Angle of tangent points s_a and s_b in different curves.

Type of Dubins curve	θ_a	θ_b
LSL	$\beta-\alpha-\pi/2$	$\beta-\alpha-\pi/2$
LSR	$\beta+\alpha-\pi/2$	$\beta+\alpha+\pi/2$
RSL	$\beta-\alpha+\pi/2$	$\beta-\alpha-\pi/2$
RSR	$\beta+\alpha+\pi/2$	$\beta+\alpha+\pi/2$

θ_a is the angle of the tangent point s_a on the starting circle, and θ_b is the angle of the tangent point s_b on the ending circle. Their values in different Dubins curves are shown in Table 1.

The tangent points $s_a(x_a, y_a)$ and $s_b(x_b, y_b)$ of the common tangent to the circles are given by

$$\begin{cases}(x_a, y_a) = (x_{c1} + r_{c1}\cos\theta_a, y_{c1} + r_{c1}\sin\theta_a)\\(x_b, y_b) = (x_{c2} + r_{c2}\cos\theta_b, y_{c2} + r_{c2}\sin\theta_b)\end{cases} \tag{8}$$

2.3. Problem formulation

The working environment of the USV in this paper is a two-dimensional plane in a lake. A two-dimensional grid map is constructed based on the grid method, and the USV path planning problem can be described as the optimal distance problem between two points. In the actual environment, there will be a certain number of obstacles between the starting point and the target point, including known obstacles and some unknown obstacles. The detector in the USV can detect the unknown obstacles within a certain range. When the USV detects the unknown obstacles, it will re-plan the path when the unknown obstacles affect the initial planned path.

To formulate the path planning problem in unknown environments clearly, we set the working environment as a configuration space. χ_{free} is the free region, and the obstacle area is represented as χ_{obs}. Then, the path planning problem in unknown environments can be expressed as follows: find a path P, and the length of path P should satisfy the following constraints:

$$P = \min(\sum_{i=1}^{n} p_i), p_i \cap \chi_{obs} = \emptyset \tag{9}$$

where path P consists of n discrete nodes $N_i \in \chi_{free}, (i = 1, 2, \dots, n)$. The nodes N_i and N_{i-1} are connected in a straight line to form the path p_i, $i \in [1, n]$. The sum of the Euclidean distances of p_i between nodes is the shortest, and the p_i should avoid obstacles.

In addition, the USV with motion constraints is considered a mathematical model for path planning. The planned paths in the grid map have a large number of turning points. In this case, the paths should be smoothed, and different smoothing methods will result in different curvatures of the paths. Assuming that the minimum turning radius of the USV with restricted movement is R_{min}, the radius of the curvature of each path p_i is C_i, and the curvature $K_i = 1/C_i$. To make the final path apply to the motion dynamics of the USV, according to the curvature formula, the smoothed paths need to satisfy the following constraints:

$$\min\sum_{i=1}^{n} K_i, \forall C_i \geq R_{min} \tag{10}$$

$$K_i = \frac{\left|p_i''\right|}{(1 + p_i'^2)^{\frac{3}{2}}} \tag{11}$$

where the total curvature of the planned path is the smallest and the path is smoother. The radius of the curvature of any section of the path should be no less than the minimum turning radius of the USV so that it can sail normally.

3. Improved D*Lite algorithm

The traditional D*Lite algorithm is a heuristic algorithm that searches from the target point to the starting point. After the initial path planning is completed, if the USV is affected by unknown obstacles on the path, it can use the previous node information for fast replanning without recalculating the heuristic values of all nodes, so that the D*Lite algorithm is well suited for path planning in unknown environments. However, the current traditional D*Lite algorithm suffers from three problems. First, the expansion range of the nodes is too large and the nodes are double-computed, which will lead to the low computational efficiency of the algorithm. Second, the grid map restricts the direction of movement of the USV, and the path length is suboptimal. In addition, the path planned by the traditional D*Lite algorithm has a large number of turning points, which is not feasible for the USV.

To solve the abovementioned problems, first, the path cost function of the traditional D*Lite algorithm is improved. The node expansion range is narrowed, and the expansion direction of nodes is limited to avoid node double computation. Then, the path cost of the grid boundary nodes is calculated by using the IDW interpolation method, and the path is no longer limited to eight directions in increments of 45°. Therefore, the number of turning points of the path is reduced, and the path length is shortened. Finally, the Dubins curve is introduced for local path smoothing to make it conform to the motion dynamics of the USV.

3.1. Reduce the range of node expansion

3.1.1. Path cost function improvement

In the traditional D*Lite algorithm, the evaluation function $k(s)$ determines the number and order of node expansions, as shown in Eq. (1), and the size of the evaluation function is mainly influenced by the path cost between nodes. To conveniently calculate the evaluation function of nodes, the path cost function between the nodes in the traditional D*Lite algorithm uses the Manhattan distance of the following form

$$c(s, s') = \max(\left|x_s - x_{s'}\right|, \left|y_s - y_{s'}\right|) \tag{12}$$

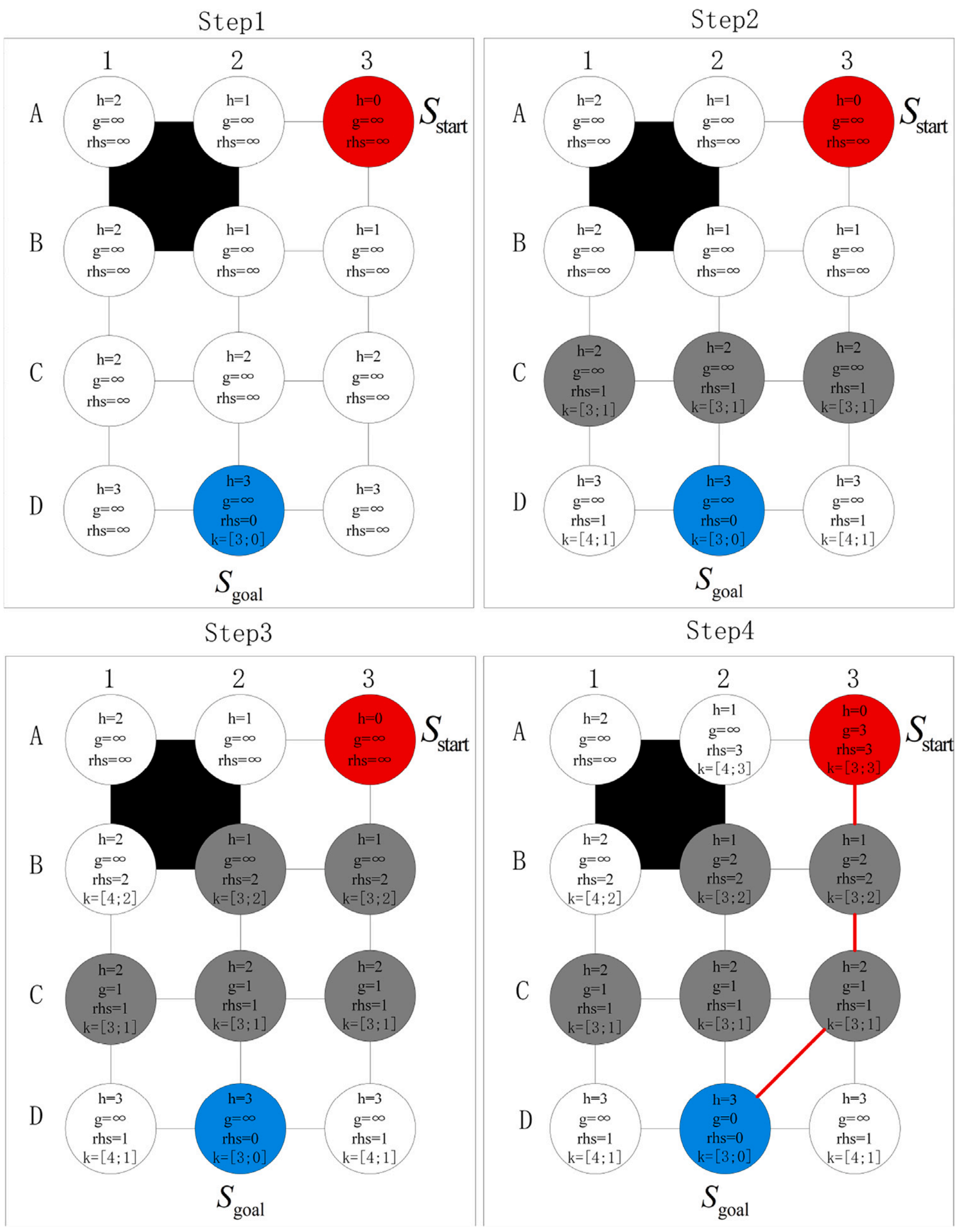

Fig. 2. Calculation process of the traditional D*Lite algorithm.

where $c(s, s')$ denotes the path cost between nodes s and s', and (x_s, y_s) and $(x_{s'}, y_{s'})$ denote the coordinates of nodes s and s'. Although this simplifies the calculation of $k(s)$, the actual calculation process will result in multiple grids with the same $k(s)$ for each expansion of nodes, generating an excessive number of expansion nodes in a complex environment. As shown in Fig. 2, in Step 1, the information of the target point is first updated so that $rhs(s) = 0$. Then, the target point is expanded to the adjacent nodes to calculate the $k(s)$ of these expanded nodes. The $k_1(s)$ values of these nodes are compared first, and then the $k_2(s)$ values are compared when the $k_1(s)$ values are equal. In Step 2, the $k(s)$ values of the three nodes C1, C2 and C3 are the smallest, and they are used as the next nodes for further expansion. In Step 3, B2 and B3, which have the smallest $k(s)$ and the same value, are used as the next expanded nodes, and so on, until they are expanded to the starting point A3, as shown in Step 4, finally forming a red path with gradually decreasing $g(s)$. In larger and more complex maps, the number of nodes to be expanded will be larger and the algorithm will be computationally inefficient.

To obtain more accurate $h(s_{start}, s)$ and $rhs(s)$, reduce the number of expansion nodes with the same evaluation function $k(s)$, and improve the computational efficiency and accuracy of the algorithm, Eq. (12) is optimized, and the optimized path cost function is as follows

$$c(s,s') = \sqrt{2}\min(|x_s - x_{s'}|, |y_s - y_{s'}|) + \| x_s - x_{s'}| - |y_s - y_{s'} \| \tag{13}$$

The effect is to define the path cost of horizontal and vertical unit movement in the grid as 1, and the path cost of diagonal unit movement as $\sqrt{2}$, which is also better suited for practical applications. As shown in Fig. 3, according to the recalculated $h(s_{start}, s)$ and $rhs(s)$, only the C2 node has the smallest $k(s)$ after the first round of expansion, and then C2 is used as the next node for expansion to B2, and finally, B2 is used for expansion until the starting point A3. It can be observed that the number of node expansions in the gray part of the optimized D*Lite algorithm is significantly reduced, which greatly improves the computing efficiency and accuracy of the algorithm.

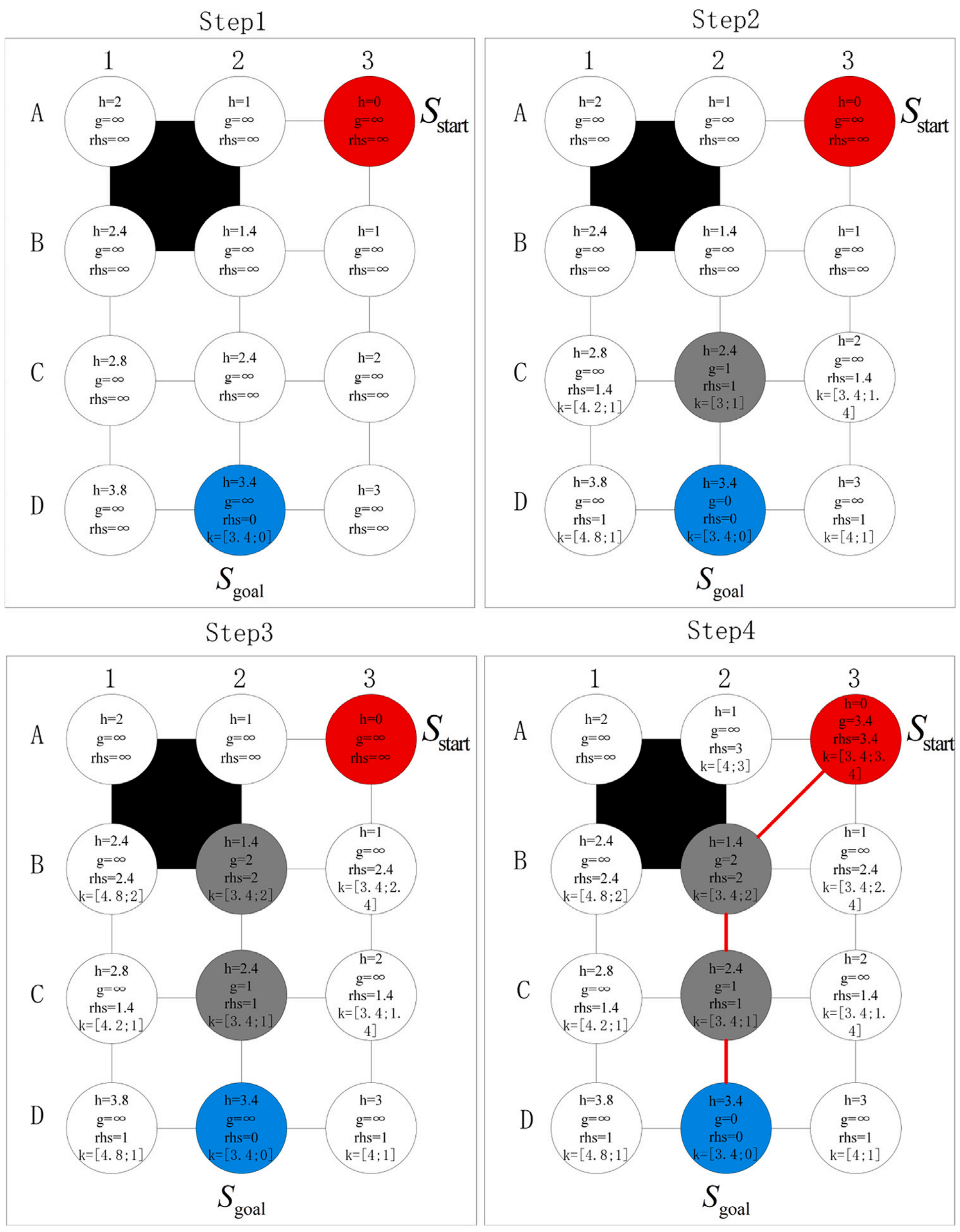
Step1
1 2 3
A B C D
h=2 g=∞ rhs=∞
h=1 g=∞ rhs=∞
h=0 g=∞ rhs=∞
S_{start}
h=2.4 g=∞ rhs=∞
h=1.4 g=∞ rhs=∞
h=1 g=∞ rhs=∞
h=2.8 g=∞ rhs=∞
h=2.4 g=∞ rhs=∞
h=2 g=∞ rhs=∞
h=3.8 g=∞ rhs=∞
h=3.4 g=∞ rhs=0 k=[3.4;0]
h=3 g=∞ rhs=∞
S_{goal}
Step2
h=2 g=∞ rhs=∞
h=1 g=∞ rhs=∞
h=0 g=∞ rhs=∞
S_{start}
h=2.4 g=∞ rhs=∞
h=1.4 g=∞ rhs=∞
h=1 g=∞ rhs=∞
h=2.8 g=∞ rhs=1.4 k=[4.2;1]
h=2.4 g=1 rhs=1 k=[3;1]
h=2 g=∞ rhs=1.4 k=[3.4;1.4]
h=3.8 g=∞ rhs=1 k=[4.8;1]
h=3.4 g=0 rhs=0 k=[3.4;0]
h=3 g=∞ rhs=1 k=[4;1]
S_{goal}
Step3
h=2 g=∞ rhs=∞
h=1 g=∞ rhs=∞
h=0 g=∞ rhs=∞
S_{start}
h=2.4 g=∞ rhs=2.4 k=[4.8;2]
h=1.4 g=2 rhs=2 k=[3.4;2]
h=1 g=∞ rhs=2.4 k=[3.4;2.4]
h=2.8 g=∞ rhs=1.4 k=[4.2;1]
h=2.4 g=1 rhs=1 k=[3.4;1]
h=2 g=∞ rhs=1.4 k=[3.4;1.4]
h=3.8 g=∞ rhs=1 k=[4.8;1]
h=3.4 g=∞ rhs=0 k=[3.4;0]
h=3 g=∞ rhs=1 k=[4;1]
S_{goal}
Step4
h=2 g=∞ rhs=∞
h=1 g=∞ rhs=3 k=[4;3]
h=0 g=3.4 rhs=3.4 k=[3.4;3.4]
S_{start}
h=2.4 g=∞ rhs=2.4 k=[4.8;2]
h=1.4 g=2 rhs=2 k=[3.4;2]
h=1 g=∞ rhs=2.4 k=[3.4;2.4]
h=2.8 g=∞ rhs=1.4 k=[4.2;1]
h=2.4 g=1 rhs=1 k=[3.4;1]
h=2 g=∞ rhs=1.4 k=[3.4;1.4]
h=3.8 g=∞ rhs=1 k=[4.8;1]
h=3.4 g=0 rhs=0 k=[3.4;0]
h=3 g=∞ rhs=1 k=[4;1]
S_{goal}

Fig. 3. Calculation process of the improved path cost function.

3.1.2. *Limit the direction of node expansion*

The traditional D*Lite algorithm will calculate the evaluation function of all surrounding successor nodes when selecting the expansion nodes. If multiple nodes have the same value of $k(s)$, their adjacent nodes may overlap. When computing the subsequent expansion nodes, these overlapping nodes will be double computed. It is easy for the USV to plan a longer path when it encounters an unknown obstacle for re-planning in a complex environment. Restricted by the motion dynamics, the USV has a maximum turning angle. If the turning angle of the re-planned path is too large, it is not feasible for the USV even though the planned path is theoretically optimal.

To solve this problem, when the traditional D*Lite algorithm calculates the $k(s)$ of the successor nodes, the direction of the calculated nodes is restricted to reduce the unnecessary calculation process. As shown in Fig. 4, the number 3 indicates that the node has been computed three times during the expansion of the adjacent nodes. However, according to the motion dynamics of USV, when the USV travels along the route from the predecessor node s_{pred} to the current node s, if the maximum turning angle is less than 135°, the node s_{succ} cannot be used

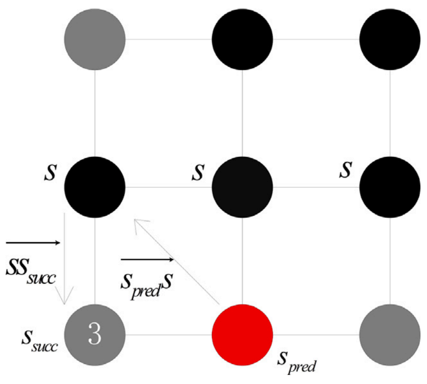

Fig. 4. Limiting the direction of node expansion.

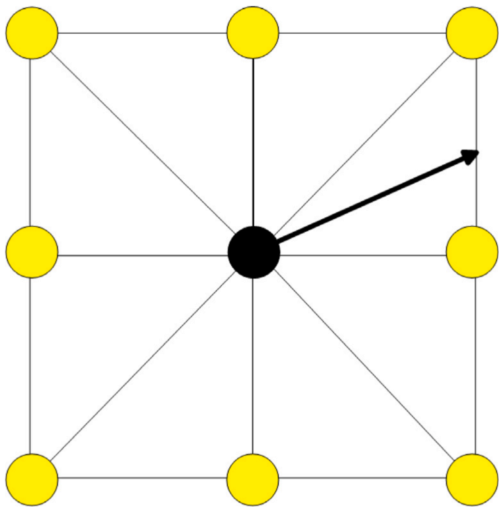

Fig. 5. Allowing paths to pass through grid boundaries.

as the next target, and there is no need to calculate it. Therefore, if ϕ is the expanded angle between vectors $\overrightarrow{s_{pred}s}$ and $\overrightarrow{ss_{succ}}$, it can be obtained from the following cosine law:

$$\phi = \arccos\left(\frac{\overrightarrow{s_{pred}s} \cdot \overrightarrow{ss_{succ}}}{\left|\overrightarrow{s_{pred}s}\right| \left|\overrightarrow{ss_{succ}}\right|}\right) \tag{14}$$

where s is the current node, s_{pred} is the predecessor node of s, and s_{succ} is the successor node of s. Assume that the maximum turning angle of the USV is φ_{max}, and the expanded node should satisfy $\varphi_{max} \geq \phi$. When calculating the $k(s)$ of adjacent nodes, if the expanded angle ϕ of a node is greater than the maximum turning angle φ_{max} of the USV, it is possible that the node has the smallest $k(s)$, but this is not necessary to calculate. Therefore, the $k(s)$ of this node is not updated during the calculation of the D*Lite algorithm, and it is directly skipped in the priority queue U and proceeds to the next cycle. After such optimization, the efficiency of the algorithm can be further improved, the number of node double calculations is reduced, and the formation of paths with large turning angles is avoided, which is more suitable for USV driving.

3.2. Increase the moveable direction

The traditional D*Lite algorithm will only count eight nodes located at the grid vertices with an angle of a multiple of 45°. With this restriction, the path can only pass through the grid vertices, and the planned path length is usually sub-optimal. Therefore, the restriction can be lifted to allow the path to pass through the grid boundaries, and the path is no longer limited to only passing through the grid vertices. As shown in Fig. 5, if the $h(s_{start}, s)$ and $rhs(s)$ of a node on the grid boundary can be obtained by interpolation, the $k(s)$ of the node can be calculated and compared with the node located at the grid vertices, and there are more choices of moving directions and shortening the path length.

For the convenience of calculation, the interpolation point selection is the same for each step. As shown in Fig. 6(a), if the current interpolation point s_{int} is at any position on the boundary, in the next calculation step, if the same interpolation point is still continued to be selected to move, the final path formed is L_1. If the grid vertex is selected to move, the path formed is L_2, and the total length of the two paths is equal. When the boundary midpoint is used as the interpolation point, as shown in Fig. 6(b), the path length is the same, but the path L_2 has one less turning point compared to L_1, and the smoothness is better. Therefore, in the actual calculation process, the interpolation point is

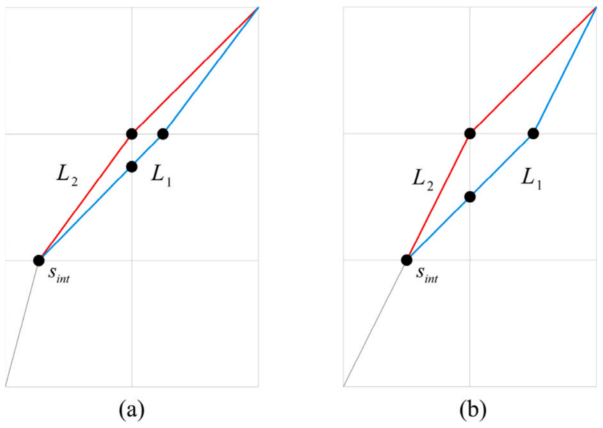

Fig. 6. Selection of interpolation points.

chosen as the midpoint of the boundary in order to reduce the turning point and shorten the path.

To calculate the evaluation function $k(s_{int})$ of the interpolated point s_{int}, the $h(s_{start}, s_{int})$ and $rhs(s_{int})$ of the point are first calculated by IDW interpolation. In the initial state, $h(s_{start}, s)$ is known for each point, and $rhs(s)$ is obtained according to Eq. (1). The adjacent grid vertices form a subset $S_{ver} = \{s_1, s_2, \dots, s_n\}$ of discrete points, which are used to calculate the interpolated weights. The weighting function is as follows

$$\omega_i = \frac{L_i^{-p}}{\sum_{i=1}^{n} L_i^{-p}} \tag{15}$$

where ω_i is the weight of each node in the subset S_{ver} of discrete points, L_i is the distance from the interpolation point s_{int} to s_i in S_{ver}, p is the power parameter, and n is the total number of discrete points, typically 8 or 6. The IDW interpolation relies heavily on the power parameter, which is a positive real number, usually taken to be between 0.5 and 3. When the power parameter is higher, the value of the interpolation point will gradually approach the value of the nearest sampling point. We find the optimal power parameter by measuring the minimum root mean square prediction error (RMSPE) with different power parameters. In the test, the minimum RMSPE of the optimal power parameter is around $p = 2$. For the convenience of the calculation, we take $p = 2$ as the default value for the experiment.

After the weights ω_i of each node s_i are obtained, the $h(s_{start}, s_{int})$ and $rhs(s_{int})$ of the interpolation point can be calculated in the same way with the following equation

$$f(s_{int}) = \sum_{i=1}^{n} \omega_i f(s_i) \tag{16}$$

where f is the $h(s_{start}, s)$ or $rhs(s)$ of the point. When the $h(s_{start}, s_{int})$ and $rhs(s_{int})$ of each interpolation point are obtained, the adjacent nodes can be calculated according to Eq. (1) and the node with the smallest $k(s)$ among them can be selected as the next node for expansion calculation. Combined with the previous optimization in Section 3.1, the final effect is shown in Fig. 7, where the gray nodes represent the $k(s)$ of the nodes have been calculated, the black nodes are the nodes with the smallest $k(s)$ after each round of expansion, and the numbers on the nodes represent the number of times the nodes have been calculated. Fig. 7(a) and (b) show the effect before and after the optimization respectively, from which it can be seen that the range of expanded nodes is reduced, the number of calculations per node is reduced, the efficiency of the algorithm is substantially improved, and the path length is shortened.

3.3. Local path smoothing

Since the D*Lite algorithm performs path planning on a grid map, the final path is still suboptimal, even though most of the turning points

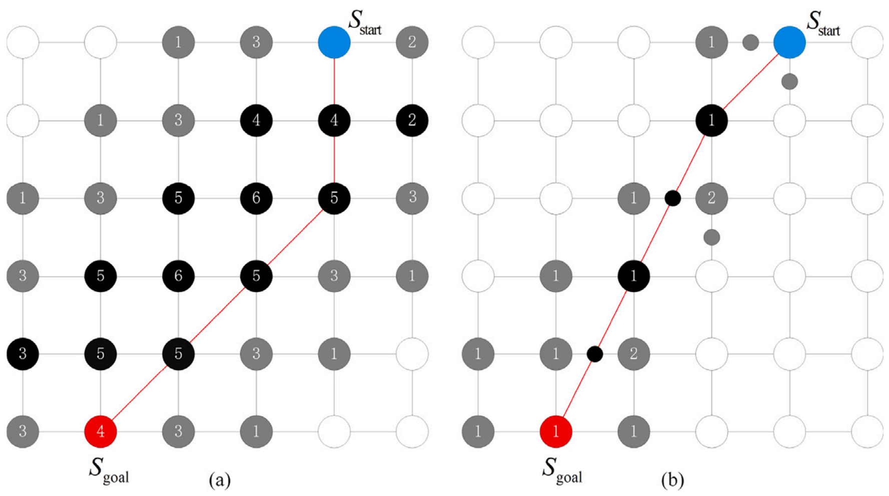

Fig. 7. Comparison of the effect of optimization.

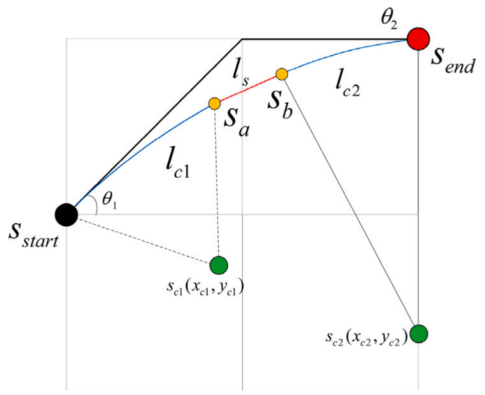

Fig. 8. Dubins curve instead of the turning point.

are reduced after the improvement in Sections 3.1 and 3.2. The paths with turning points are not suitable for constant-speed USV; otherwise, the USV needs to stop at the water surface for steering, which will waste time. The path planning in the unknown environment relies on the real-time detection of sensors, which cannot smooth the global path. Therefore, this paper combines the D*Lite algorithm with the Dubins curve to perform local smoothing in the search process of the D*Lite algorithm.

As shown in Fig. 8, after the initial path is planned by the D*Lite algorithm, the nearest grid nodes $s_{start}(x_1, y_1)$ and $s_{end}(x_2, y_2)$ before and after each turning point are obtained, as well as the directional angles θ_1 and θ_2. After the turning radius of the USV is set, the center of the starting circle $s_{c1}(x_{c1}, y_{c1})$ and the center of the ending circle $s_{c2}(x_{c2}, y_{c2})$ are obtained according to Eq. (5), and the coordinates of $s_a(x_a, y_a)$ and $s_b(x_b, y_b)$ are obtained according to Table 1 and Eq. (8). The nodes s_a and s_b are connected to form the line segment S in the Dubins curve, and the arcs l_{c1} and l_{c2}, with s_{c1} and s_{c2} as the centers, are the arc segments C respectively. Finally, these three paths form the complete Dubins curve instead of the turning point.

When the initial planning of the path is completed, all turning points are replaced by LSL or RSR curves. When an unknown obstacle is detected in the way of the USV, the USV can still drive to the nearest node on the smoothed path and can then accurately perform the path replanning operation because of the safety distance, and the first turning point on the new path after replanning is replaced by the LSR or RSL curve.

3.4. Algorithm overview

In summary, the specific process of the improved D*Lite algorithm proposed in this paper for USV path planning in an unknown environment is as follows. First, the starting point and the target point are set on the grid map generated based on the grid method, and the parameters of the nodes are initialized according to the improved path cost function. Second, the nodes are gradually expanded from the target point, and IDW is used to obtain the parameters of the grid boundary nodes, and the node with the smallest valuation function is selected until it is expanded to the starting point. Third, after obtaining the initial path, the Dubins algorithm is introduced to locally smooth the turning points. Finally, a smooth collision-free path is obtained from the starting point to the target point, which conforms to the motion dynamics of the USV.

Algorithm 1 The improved D*Lite algorithm.

```
Step 1 Input grid map information
Step 2 Set the starting point $s_{start}$ and target point $s_{goal}$ of the USV
Step 3 Expand adjacent nodes from $s_{goal}$
Step 4 Calculate the $k(s)$ of adjacent grid vertices
Step 5 Calculate the $k(s)$ of adjacent grid boundary nodes with IDW
Step 6 Compare $k(s)$ and select the node with the smallest $k(s)$ as the next expanded node
Step 7 Expand the nodes constantly until $s_{start}$ is reached
Step 8 Find the initial path based on the minimum $g(s)$ from the starting point
Step 9 The nodes adjacent to each turning point are used as the starting and ending points of the Dubins curves
Step 10 Set the minimum turning radius for the USV
Step 11 Calculate the Dubins curves and replace the folded segments

Step 12 The USV moves to the node with the smallest $g(s)$
Step 13 **If** the surrounding environment has not changed
Step 14     update the adjacent nodes and return to Step 2
Step 15 **else** the current node is the new starting point $s'_{start}$ and movement is continued
Step 16 **If** node $s'_{start}$ is node $s_{goal}$
Step 17     Complete path planning between the starting and target points
Step 18 **else** return to Step 11
```

4. Simulation experiments

To verify the advantages of the improved D*Lite algorithm, several simulation experiments were performed. Windows 10 was used as

Table 2
Parameters of the USV.

Symbol	Definition	Numerical value
L (m)	Length of the ship	1.5
W (m)	Width of the ship	0.9
R_{min} (m)	Minimum turning radius	1.2
φ_{max} (deg)	Maximum turning angle	60
D_{safe} (m)	Safe distance	2.5
v	Ship speed	2

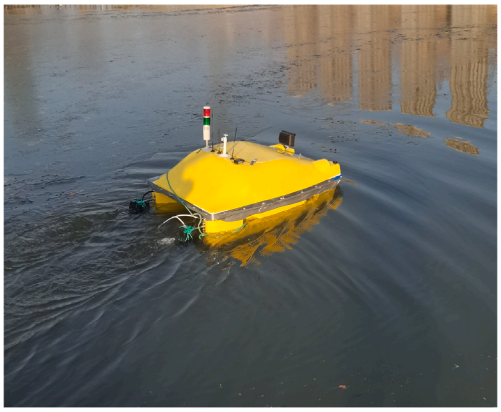
Fig. 9. The physical object of the USV in the experiment.

the operating system and MATLAB R2018b as the simulation tool. The hardware platform was an Intel Core i5-10400 processor with a frequency of 2.9 GHz and a memory of 16 GB. First, the path planning results of the USV with different algorithms in the same known environment were compared. Second, simulations were performed in an environment with unknown obstacles to demonstrate the effectiveness of the improved D*Lite algorithm for application in USVs.

The physical object of the USV used for the experiment is shown in Fig. 9. This research used real water areas as the experimental environments. The obstacles in the waters included reefs, buoys, and groups of water grass. We used a six-rotor UAV to take aerial photographs of the water areas. The aerial photographs were converted into simulated environmental information through image processing technology, and then, the environmental information was transformed into a two-dimensional grid map using the grid method. The black grid represents the obstacle, and the white grid represents the passable area. When the obstacle area is less than one grid size, it is complemented by a grid. Some areas in the photos were difficult to identify by image processing because of the camera resolution that limited image sharpness, and some obstacles were hidden beneath the surface of the water. Therefore, the detection of these unknown obstacles required the lidar sensor of the USV. The lidar sensor detection accuracy decreases with increasing distance during water surface point cloud acquisition. The obstacle detection range of the USV is 20 m according to the performance of the lidar sensor. The lidar sensor obtains the position and shape of unknown obstacles by emitting multiple laser beams. The unknown obstacles are then transformed into gray grids by the grid method in the map. To avoid collision with obstacles, the safety distance is set in the algorithm according to the size of the USV; so, the USV can avoid obstacles during the turning or re-planning process when detecting an unknown obstacle.

A 20 × 20 grid map model was created and the scale of the map is 4 m. The parameters of the USV in the simulation experiment are shown in Table 2, and the simulation parameters of the improved D*Lite algorithm are shown in Table 3.

Table 3
Simulation parameters of the improved D*Lite algorithm.

Symbol	Definition	Numerical value
U	Priority queue	$\emptyset$
k_m	Key modifier	0
$rhs(s)$	Path cost of the node s	∞
$rhs(s_{goal})$	Path cost of the node s_{goal}	0
$g(s)$	Actual path cost of node s	∞
p	Power parameter	2

Table 4
Performance comparison of the three algorithms in known environments.

Performance indicator	D*Lite	Improved D*Lite	Improved D*Lite with Dubins curve
Planning time (s)	0.4351	0.2145	**0.2950**
Planning length (m)	111.1964	106.2080	**106.4708**
Number of node calculations	845	217	**217**
Number of path turning points	14	9	**0**
Cumulative turning angle (°)	630	200.61	**0**

4.1. Path planning in known obstacle environments

First, the traditional D*Lite algorithm and the improved D*Lite algorithm proposed in this paper were simulated in the same map where all obstacles were known, and 50 simulations were performed in this scenario to avoid the contingencies of the experiments. The task of the USV is to plan a path from the starting point (1,1) to the target point (19,19), and the planning time, path length, number of node calculations, number of turning points, and cumulative turning angle were used as evaluation indicators.

The simulation results are shown in Figs. 10–11, where the blue line segment is the planning path, and the green part represents the expanded nodes. Compared with the traditional D*Lite algorithm, the improved D*Lite algorithm requires a significantly less expanded node range in the calculation process. The specific performance comparison of the algorithm is shown in Table 4, and the average values were taken for each indicator. In terms of planning efficiency, the planning time of the improved D*Lite algorithm decreased from 0.4351 s to 0.2145 s. The number of calculations for all nodes in the expansion process was reduced from 845 to 217, and this indicates that the efficiency of the improved D*Lite algorithm was improved by 74.32%. Since different computer configurations can lead to slight deviations in planning times, but the number of node computations of the algorithm is fixed on the same map, this indicator is more intuitive for comparing the planning efficiency of the algorithm. Even though the application of IDW expands the range of nodes in each expansion process and increases the amount of computation in a single step, the overall amount of computation is still significantly reduced when combined with the subsequent optimization process; so, the planning efficiency of the algorithm is still greatly improved compared to that of the traditional D*Lite algorithm.

In terms of path quality, the planning length of the improved D*Lite algorithm (106.2080 m) is shorter than that of the traditional D*Lite algorithm (111.1964 m), the path turning points are reduced from 14 to 9, and the cumulative turning angle of the USV is reduced from 630° to 200.61°, which makes the optimized path more applicable even when applied to unmanned vehicles or UAVs. Then, combined with Dubins algorithm for local path smoothing, the final simulation effect is shown in Fig. 12. Since the algorithm needs to ensure that the smoothed path still passes through the grid nodes so that the USV can accurately replan from the grid nodes when it detects an unknown obstacle, the Dubins algorithm will sacrifice part of the shortest path when there are continuous turning points on the initial path. From Table 4, it can be seen that the smoothed path length is 106.4708 m, which is slightly longer than that of the improved D*Lite algorithm (106.2080 m), but the overall path no longer has turning points, which greatly improves

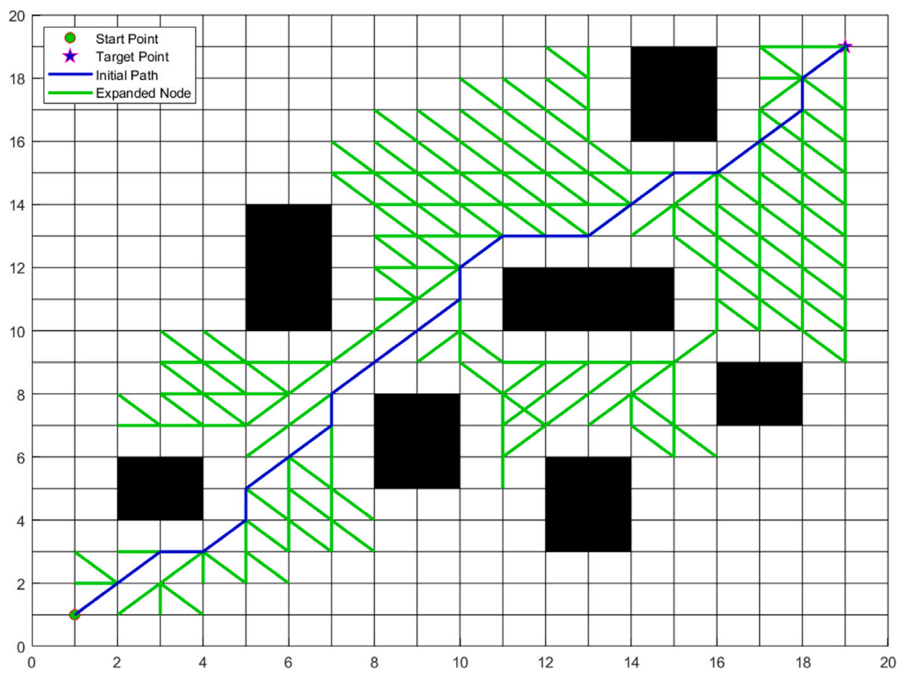

Fig. 10. Traditional D*Lite algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

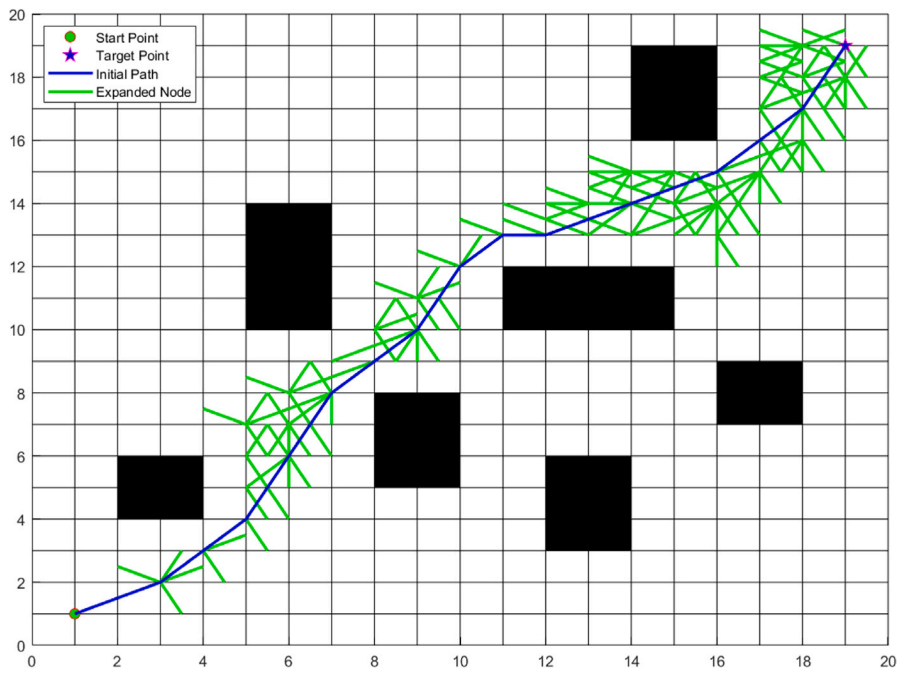

Fig. 11. Improved D*Lite algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the movement efficiency of the USV traveling at constant speed and reduces the energy consumption, and such a path is more suitable for the USV to travel.

To avoid the contingency of the experiment, the experiments were conducted separately in this scenario with different starting and target points, and the results are shown in Table 5. In addition, the experiments were conducted separately on two other maps, noted as Cases 1–2, where the obstacles were randomly generated with starting point (1,1) and target point (19,19), and the performance of each algorithm in these two environments is shown in Table 6. From Tables 5 and 6, it can be seen that the algorithm proposed in this paper is always better than the traditional D*Lite algorithm in terms of planning length, planning time, and turning angle in various cases, indicating that the algorithm in this paper has stronger applicability and higher performance.

4.2. Path planning in unknown obstacle environments

To test the performance of the algorithm in an unknown obstacle environment, unknown obstacles were added to the path traveled by the USV in the same scenario as above and replaced by a gray grid. The simulation parameters were the same as in the previous experiments. When the USV was initially planned, the unknown obstacles were still treated as passable area, and the USV can be replanned in advance if

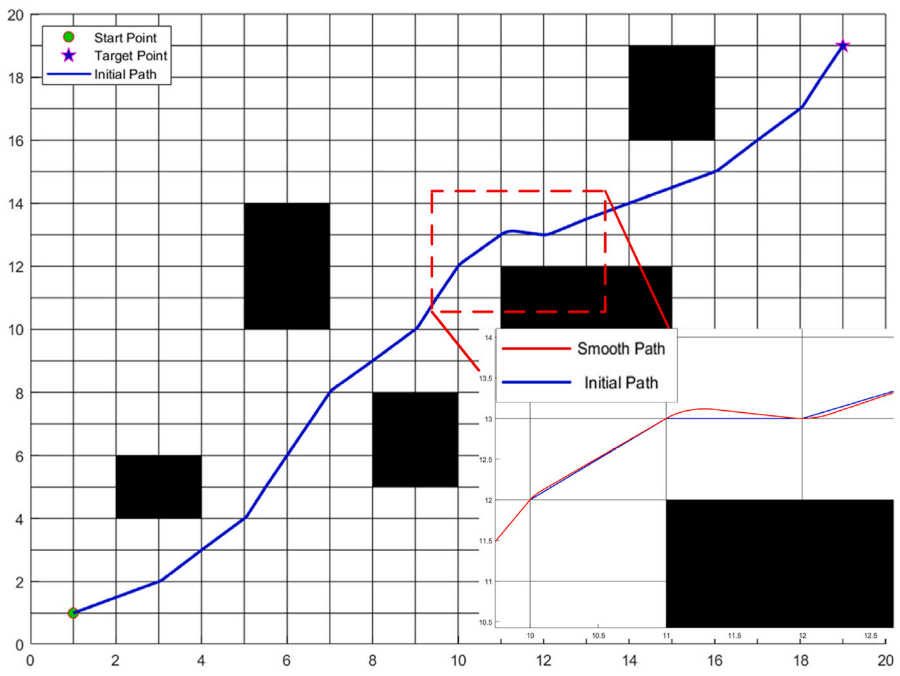

Fig. 12. Improved D*Lite algorithm with Dubins curve.

Table 5
Comparison of path quality under different starting and target points.

Start points	Target point	Performance indicator	D*Lite	Improved D*Lite	Improved D*Lite with Dubins curve
(3,17)	(18,4)	Planning length (m)	86.2252	81.4358	**81.9514**
		Planning time (s)	0.4265	0.2118	**0.2684**
		Turning angle (°)	315	110.61	**0**
(1,12)	(19,3)	Planning length (m)	86.9168	83.7854	**84.0612**
		Planning time (s)	0.4234	0.2085	**0.2473**
		Turning angle (°)	225	126.87	**0**
(3,17)	(18,4)	Planning length (m)	95.8824	91.4634	**91.6068**
		Planning time (s)	0.4398	0.2542	**0.2885**
		Turning angle (°)	450	163.74	**0**

Table 6
Statistical result analysis of the three algorithms in Cases 1–2.

Case	Algorithm	Planning length (m)	Planning time (s)	Turning angle (deg)
1	D*Lite	118.2252	0.4232	630
	Improved D*Lite	113.6027	0.2285	351.87
	Improved D*Lite with Dubins curve	**113.9352**	**0.2726**	**0**
2	D*Lite	113.5492	0.3965	540
	Improved D*Lite	109.8685	0.2392	306.87
	Improved D*Lite with Dubins curve	**110.5688**	**0.2621**	**0**

the unknown obstacles are detected to affect its path during the driving along the initial path.

The final simulation results are shown in Figs. 13–15. The blue line segment represents the initial planned path, while the red line segment represents the re-planned path after the USV detects an unknown obstacle. Each algorithm was subjected to 50 simulation experiments. The specific performance comparison of the algorithms is shown in Table 7, and the average values were obtained for each indicator. In the scenario with the unknown environment, it can be seen from Table 7 that the indicators of the improved D*Lite algorithm still outperformed those of the traditional D*Lite algorithm. Since the scenario was the same, the initial planning time was basically the same as that in the previous experiments, but the time required for the replanning process was significantly reduced, with the average time shortened from 0.2835 s to 0.1232 s. The number of node calculations (including the replanning process) was reduced from 1477 to 370, and the overall computational efficiency was improved by 74.95%. It can be seen from the local comparison chart that when the Dubins curve was used to smooth the path in real time, the USV could still complete the replanning process accurately from the node in advance when it detected an unknown obstacle; so, the USV could move smoothly even in the replanning process. This avoided the disadvantage that the path of an unknown environment that is not smooth is not suitable for the actual application of the USV, which shows that the improved D*Lite algorithm has a better performance in the environment with the unknown scenario and also has strong applicability. The path length after using the Dubins curve was 115.0896 m, which is still better than that of the traditional D*Lite algorithm (116.8528 m). As in the previous experiments, the quality of the smoothed path was significantly improved by eliminating all turning points of the path, making the path more suitable for working with USVs in water environments.

5. Conclusion

In this paper, an improved D*Lite algorithm is proposed for USV path planning in unknown environments. It solves the problems of excessive node expansion range and double node calculation of the traditional D*Lite algorithm and improves the computational efficiency of the algorithm. The IDW interpolation method was used to eliminate

Table 7
Performance comparison of the three algorithms in unknown environments.

Performance indicator	D*Lite	Improved D*Lite	Improved D*Lite with Dubins curve
Planning time (s)	0.4351	0.2145	**0.2950**
Re-planning time (s)	0.2835	0.1232	**0.1535**
Planning length (m)	111.1964	106.2080	**106.4708**
Re-planning length (m)	116.8528	112.8176	**115.0896**
Number of node calculations	845	217	**217**
Number of re-planning nodes calculations	632	153	**153**
Number of path turning points	14	9	**0**
Number of re-planning path turning points	16	11	**0**
Cumulative turning angle (°)	630	200.61	**0**
Re-planning path cumulative turning angle (°)	720	396.87	**0**

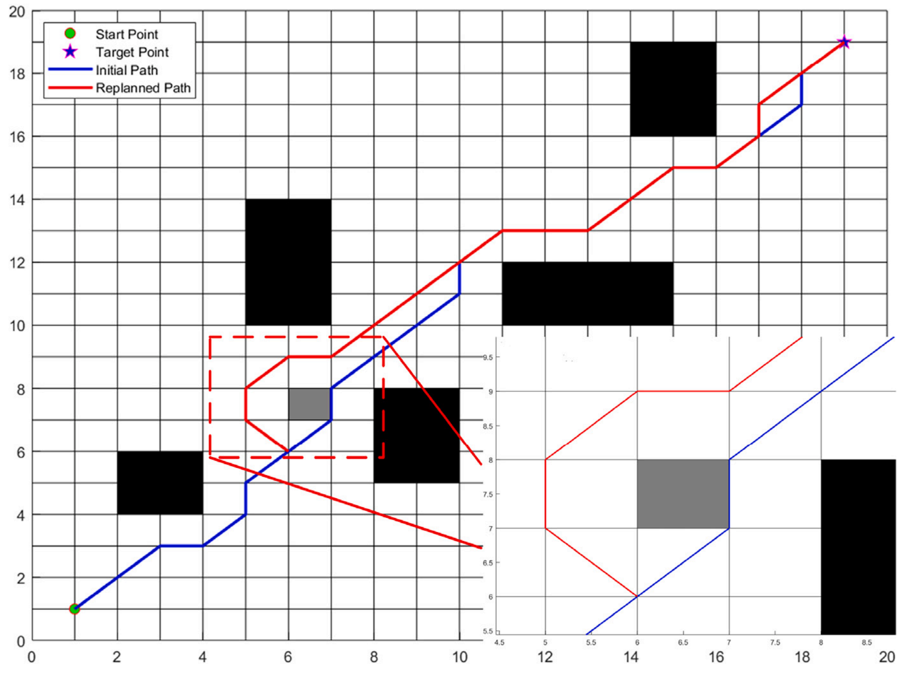

Fig. 13. Traditional D*Lite algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

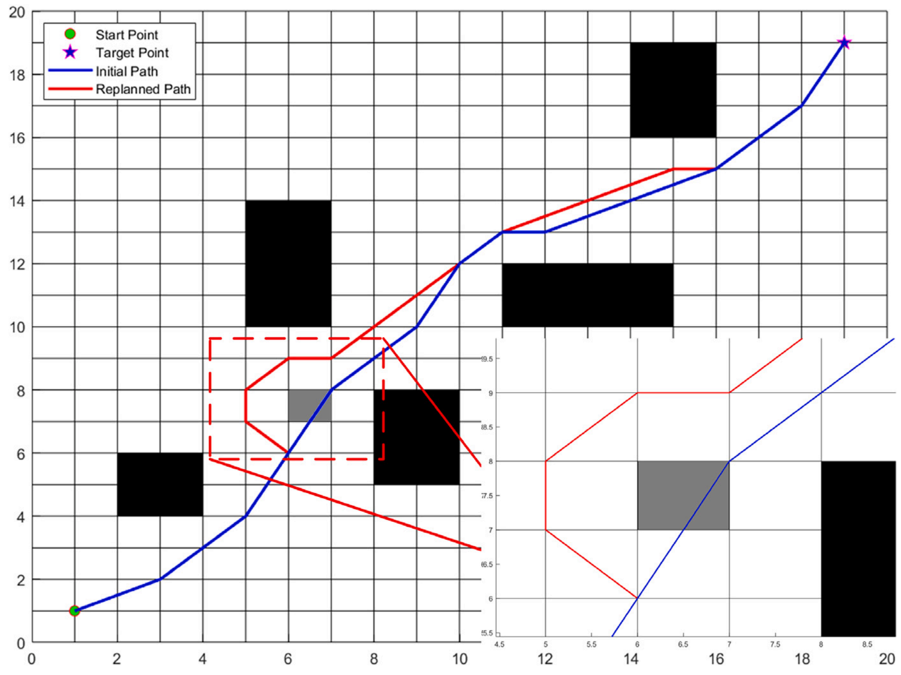

Fig. 14. Improved D*Lite algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

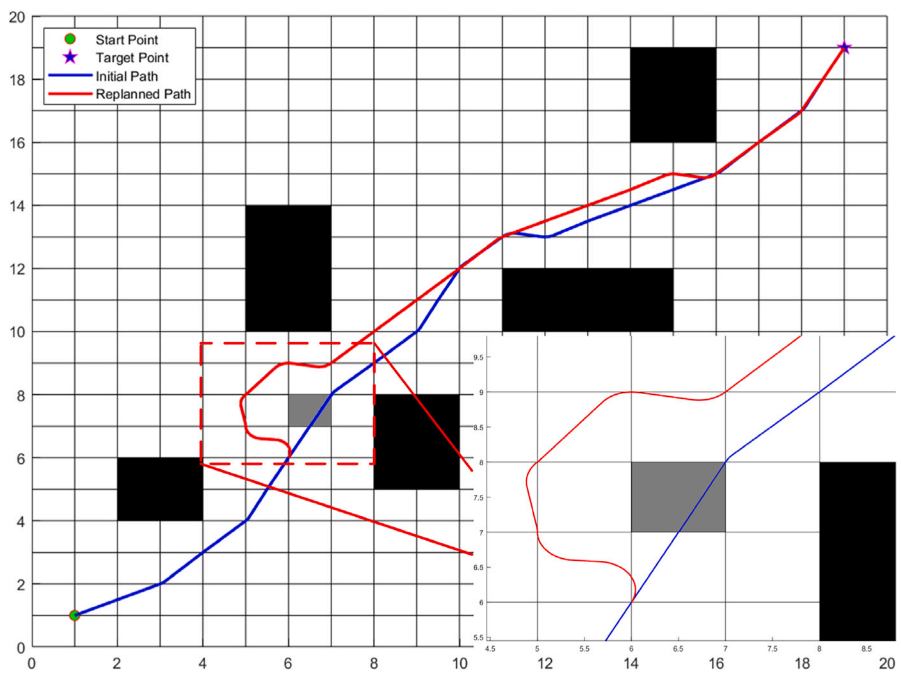

Fig. 15. Improved D*Lite algorithm with Dubins curve. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the limitation of the grid map and shorten the path length. The Dubins curve was introduced for local path smoothing to obtain a smooth collision-free path that conformed to the motion dynamics of USV. To verify the effectiveness and applicability of the algorithm, simulation experiments were performed in known environments and unknown environments. The simulation results showed that, in terms of the planning time, planning length, path smoothing, and other indicators, the proposed algorithm has obvious advantages over the traditional D*Lite algorithm. The planned paths were more suitable for USVs to travel in lake environments.

In future research, more complex lake environments will be constructed, and more external factors will be considered for their effects on USV driving, such as moving obstacles, water flow, and wind direction, and the single target point path planning will be extended to multiple target points.

CRediT authorship contribution statement

Jiabin Yu: Conceptualization, Formal analysis, Writing – original draft. **Meng Yang:** Conceptualization, Formal analysis, Simulation experiments, Writing – original draft. **Zhiyao Zhao:** Methodology, Funding acquisition. **Xiaoyi Wang:** Writing – review & editing, Funding acquisition. **Yuting Bai:** Writing – review & editing. **Jiguang Wu:** Formal analysis, Simulation experiments. **Jiping Xu:** Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported in part by the Beijing Excellent Talent Training Support Project for Young Top-Notch Team (No. 20180000-26833TD01), Outstanding Youth Cultivation Project of Beijing Technology and Business University.

References

Abubakr, O.A., Jaradat, M.A.K., Hafez, M.A., 2018. A reduced cascaded fuzzy logic controller for dynamic window weights optimization. In: 2018 11th International Symposium on Mechatronics and its Applications. ISMA, IEEE, pp. 1–4.
Ali, H., Gong, D., Wang, M., Dai, X., 2020. Path planning of mobile robot with improved ant colony algorithm and MDP to produce smooth trajectory in grid-based environment. Front. Neurorobot. 14, 44.
Bashiri, M., Rezanezhad, M., Tavakkoli-Moghaddam, R., Hasanzadeh, H., 2018. Mathematical modeling for a p-mobile hub location problem in a dynamic environment by a genetic algorithm. Appl. Math. Model. 54, 151–169.
Chao, N., Liu, Y.-k., Xia, H., Peng, M.-j., Ayodeji, A., 2019. DL-RRT* algorithm for least dose path Re-planning in dynamic radioactive environments. Nucl. Eng. Technol. 51 (3), 825–836.
Cil, Z.A., Mete, S., Serin, F., 2020. Robotic disassembly line balancing problem: A mathematical model and ant colony optimization approach. Appl. Math. Model. 86, 335–348.
Cui, P., Yan, W., Cui, R., Yu, J., 2018. Smooth path planning for robot docking in unknown environment with obstacles. Complexity 2018.
Dabiri, N., Tarokh, M.J., Alinaghian, M., 2017. New mathematical model for the bi-objective inventory routing problem with a step cost function: A multi-objective particle swarm optimization solution approach. Appl. Math. Model. 49, 302–318.
Dijkstra, E.W., et al., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.
Dubins, L.E., 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. Amer. J. Math. 79 (3), 497–516.
Feng, L., Zhou, M., Hu, B., 2020. A hybrid motion planning algorithm for multi-robot formation in a dynamic environment. In: 2020 IEEE 16th International Conference on Automation Science and Engineering. CASE, IEEE, pp. 343–348.
Ferguson, D., Stentz, A., 2005. The Field D* Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments. Tech. Rep. CMU-RI-TR-05-19, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.
Ghadiry, W., Habibi, J., Aghdam, A.G., Zhang, Y., 2020. Optimal path tracking with Dubins' vehicles. IEEE Syst. J. 15 (1), 466–477.
Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), 100–107.

Huang, L., Zhou, F.T., 2020. Mobile robot path planning based on path optimization D*Lite algorithm. J. Control Decis. 35 (4), 877.

Jose, S., Antony, A., 2016. Mobile robot remote path planning and motion control in a maze environment. In: 2016 IEEE International Conference on Engineering and Technology. ICETECH, IEEE, pp. 207–209.

Kavraki, L.E., Svestka, P., Latombe, J.-C., Overmars, M.H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. 12 (4), 566–580.

Koenig, S., Likhachev, M., 2005. Fast replanning for navigation in unknown terrain. IEEE Trans. Robot. 21 (3), 354–363.

Koenig, S., Likhachev, M., Furcy, D., 2004. Lifelong planning A*. Artificial Intelligence 155 (1–2), 93–146.

LaValle, S.M., et al., 1998. Rapidly-exploring random trees: A new tool for path planning.

Le, A.T., Bui, M.Q., Le, T.D., Peter, N., 2017. D*Lite with reset: Improved version of D* lite for complex environment. In: 2017 First IEEE International Conference on Robotic Computing. IRC, IEEE, pp. 160–163.

Lin, N., Tang, J., Li, X., Zhao, L., 2019. A novel improved bat algorithm in UAV path planning. J. Comput. Mater. Contin. 61, 323–344.

Maurović, I., Seder, M., Lenac, K., Petrović, I., 2017. Path planning for active SLAM based on the D* algorithm with negative edge weights. IEEE Trans. Syst. Man Cybern.: Syst. 48 (8), 1321–1331.

Mohanty, P.K., Parhi, D.R., 2016. Optimal path planning for a mobile robot using cuckoo search algorithm. J. Exp. Theor. Artif. Intell. 28 (1–2), 35–52.

Nascimento, L.B., Pereira, D.S., Alsina, P.J., Silva, M.R., Fernandes, D.H., Roza, V.C., Sanca, A.S., 2018. Goal-biased probabilistic foam method for robot path planning. In: 2018 IEEE International Conference on Autonomous Robot Systems and Competitions. ICARSC, IEEE, pp. 199–204.

Osmankovic, D., Tahirovic, A., Magnani, G., 2017. All terrain vehicle path planning based on d* lite and MPC based planning paradigm in discrete space. In: 2017 IEEE International Conference on Advanced Intelligent Mechatronics. AIM, IEEE, pp. 334–339.

Pattnaik, S., Mishra, D., Panda, S., 2022. A comparative study of meta-heuristics for local path planning of a mobile robot. Eng. Optim. 54 (1), 134–152.

Przybylski, M., Putz, B., 2017. D*Extra Lite: a Dynamic A* with search-tree cutting and frontier-gap repairing. Int. J. Appl. Math. Comput. Sci. 27 (2).

Qi, J., Yang, H., Sun, H., 2020. MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment. IEEE Trans. Ind. Electron. 68 (8), 7244–7251.

Ramalingam, G., Reps, T., 1996. An incremental algorithm for a generalization of the shortest-path problem. J. Algorithms 21 (2), 267–305.

Reyes, N.H., Barczak, A.L., Susniak, T., 2018. Autonomous navigation in partially known confounding maze-like terrains using D*Lite with poisoned reverse. In: 2018 World Symposium on Digital Intelligence for Systems and Machines. DISA, IEEE, pp. 67–76.

Rösmann, C., Feiten, W., Wösch, T., Hoffmann, F., Bertram, T., 2012. Trajectory modification considering dynamic constraints of autonomous robots. In: ROBOTIK 2012; 7th German Conference on Robotics. VDE, pp. 1–6.

Saranrittichai, P., Niparnan, N., Sudsang, A., 2013. Robust local obstacle avoidance for mobile robot based on dynamic window approach. In: 2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology. IEEE, pp. 1–4.

Sebastian, B., Ben-Tzvi, P., 2019. Physics based path planning for autonomous tracked vehicle in challenging terrain. J. Intell. Robot. Syst. 95 (2), 511–526.

Stentz, A., 1993. Optimal and efficient path planning for unknown and dynamic environments. In: International Journal of Robotics and Automation. Citeseer.

Vautier, U., Viel, C., Wan, J., Jaulin, L., Hone, R., Dai, M., 2019. Restricted orientation dubins path with application to sailboats. IEEE Robot. Autom. Lett. 4 (4), 4515–4522.

Wang, J., Shang, X., Guo, T., Zhou, J., Jia, S., Wang, C., 2019. Optimal path planning based on hybrid genetic-cuckoo search algorithm. In: 2019 6th International Conference on Systems and Informatics. ICSAI, IEEE, pp. 165–169.

Yaoming, Z., Yu, S., Anhuan, X., Lingyu, K., 2021. A newly bio-inspired path planning algorithm for autonomous obstacle avoidance of UAV. Chin. J. Aeronaut. 34 (9), 199–209.

Yoo, Y., 2019. Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches. Knowl.-Based Syst. 178, 74–83.

Yuan, H., Zhang, G., Li, Y., Liu, K., Yu, J., 2019. Research and implementation of intelligent vehicle path planning based on four-layer neural network. In: 2019 Chinese Automation Congress. CAC, IEEE, pp. 578–582.

Zhu, Z., Xie, J., Wang, Z., 2019. Global dynamic path planning based on fusion of a* algorithm and dynamic window approach. In: 2019 Chinese Automation Congress. CAC, IEEE, pp. 5572–5576.