BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 72(2), 2024, Article number: e148834
DOI: 10.24425/bpasts.2024.148834

A global path-planning algorithm based on critical point diffusion binary tree for a planar mobile robot

Zhiyong YANG, Lipeng WANG*, Zejun CAO, Zhi ZHANG, and Zhuang XU

College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin, 150001, China

Abstract. A global path-planning algorithm for robots is proposed based on the critical-node diffusion binary tree (CDBT), which solves the problems of large memory consumption, long computing time, and many path inflection points of the traditional methods. First of all, the concept of Quad-connected, Tri-connected, Bi-connected nodes, and critical nodes are defined, and the mathematical models of diverse types of nodes are established. Second, the CDBT algorithm is proposed, in which different planning directions are determined due to the critical node as the diffusion object. Furthermore, the optimization indices of several types of nodes are evaluated in real-time. Third, a path optimization algorithm based on reverse searching is designed, in which the redundant nodes are eliminated, and the constraints of the robot are considered to provide the final optimized path. Finally, on one hand, the proposed algorithm is compared with the A* and RRT methods in the ROS system, in which four types of indicators in the eight maps are analysed. On the other hand, an experiment with an actual robot is conducted based on the proposed algorithm. The simulation and experiment verify that the new method can reduce the number of nodes in the path and the planning time and is suitable for the motion constraints of an actual robot.

Keywords: robot; global path; rapid planning; critical point; reverse optimization.

1. INTRODUCTION

The planar mobile robot is a complex system that integrates perception, decision-making, planning, control, and other functions. Path planning is an indispensable part of the planar mobile robot applications [1,2]. Excellent planning algorithms improve the operational efficiency of the robot and provide a solid foundation for subsequent decision-making and control [3–5]. In planar mobile robot path-planning tasks, the grid map is the primary representation, which greatly reduces the complexity of path planning by dividing the map into uniform rectangular blocks [6–8].

Path-planning algorithm is an important part of robot navigation, mainly referring to the automatic planning of a path from the starting point to the target point within the corresponding area. In this process, it is necessary to ensure no collision occurs and the cost of pathfinding is low. When conducting path planning, we need to consider the acquisition of the starting and ending positions, the environmental representation of obstacles, planning methods, and search methods. The objective of the path-planning problem is to find the shortest, fastest, or most economical path considering various constraints, to fulfil specific requirements. Global path planning refers to the process of finding the optimal path from a starting point to a target point, considering the constraints of the environment. Global path planning can help robots quickly find the optimal path in complex environments, avoiding situations where robots get lost or take detours in maze-like environments, thereby improving the navigation efficiency of robots.

The major algorithms of global path planning include Dijkstra, A*, LPA*, D*, D* Lite, etc. [9–14]. LPA*, D*, and D* Lite are mainly used to complete path planning under dynamic conditions. A* algorithm is often used to deal with the static environment. In recent years, many scholars have improved the algorithm above from different viewpoints. For example, Song *et al.* [15] have improved the heuristic function of A* to make it closer to the actual distance between the start and to target. Yang *et al.* [16] combined A* with a bacterial foraging optimization algorithm to improve planning efficiency. Kim *et al.* [17] proposed a ray casting and tracking, which radiates limited light from the starting node, and reflects light through the collision with obstacles until reaching the target area. Zhang *et al.* [18] proposed two-phase A* with the adaptive heuristic weights, which generates an approximately optimal global path, effectively.

The DB-CNN proposed by Zhang *et al.* [19] combined deep reinforcement learning with a path-planning mission, which estimates the value function through the neural network to complete the global path planning. Xia *et al.* [20] improved the quantum ant colony algorithm, which is an algorithm that benefits from the high efficiency of quantum computing and the optimization ability of the ant colony algorithm. Huang *et al.* [21] improved the particle swarm optimization method. The final candidate path competes with the overall best candidate path based on the standard particle swarm optimization, which improved the search speed and avoided local minimum. Luan *et al.* [22] applied a genetic algorithm to path planning and a combined memory algorithm to solve the global path-planning task of a differential wheeled robot.

*e-mail: wlp_heu@163.com

Manuscript submitted 2023-02-25, revised 2023-09-07, initially accepted for publication 2023-10-08, published in March 2024.

Yu *et al.* [23] designed the NPQ-RRT* algorithm, in which the attitude of the robot is combined to adjust the angle of the next planning step. Song *et al.* [24] propose a real-time obstacle avoidance decision model based on machine learning algorithms, an improved smooth rapidly exploring random tree algorithm, and an improved hybrid genetic algorithm-ant colony optimization. Wang *et al.* [25] designed a multi-heuristic strategy for RRT and optimized the multiple modules such as collision detection, goal-biased guidance, bidirectional extension, goal-point attempt, and branch pruning. Pazderski [26] utilized a motion controller and a navigation velocity field to plan the path of the robot.

Among the methods mentioned above, the RRT series methods are not suitable for maps with narrow spaces, and their operational efficiency will be extremely low on such maps. The improved particle swarm and genetic algorithms are prone to falling into local minima, which prevents them from finding the optimal path under certain complex conditions. Furthermore, although the improved A* algorithm with a heuristic function can slightly enhance search efficiency, it does not significantly reduce search time.

A method for global path planning in dense environments based on a critical node diffusion binary tree is proposed to address the problems in global path planning in complex environments. First, the node types in the grid map are classified into four categories: Bi-connected nodes, Tri-connected nodes, and Quad-connected nodes. Second, the entire path-planning process is divided into two stages: the search stage and the diffusion stage. In the search stage, the current node is searched along a certain direction until the node type of the current node changes, and then it transitions to the diffusion stage. In the diffusion stage, the diffusion direction for the next search stage is determined based on the critical nodes obtained from the previous search. Third, after finding the path, the path is reversed by selecting the parent nodes to ensure the smoothness of the global path. Finally, the turning radius is minimized to smooth the path, and the stability of the robot movement is improved.

The sections of this paper are arranged as follows: The concept and the model of critical nodes are proposed in Section 2. The path optimization method based on the principle of critical node diffusion binary tree is introduced in Section 3. The compared simulations and actual experiments are described in Section 4. Section 5 concludes the whole paper.

2. MAP DESCRIPTION

2.1. Grid map description

A grid map is a map represented by a large number of grids composed of cells. For a two-dimensional space, the size of the grid in the grid map is only determined by the dimensions of the map. The grid map is shown in Fig. 1a. The blank grids are the obstacles, and the white grids are the access area for the robot.

When the robot is in the blank area, the walking direction is shown in Fig. 1b. The red dot represents the current position of the robot, and the arrow represents the direction, where the robot can move. When there is no obstacle, the robot can move in eight directions.

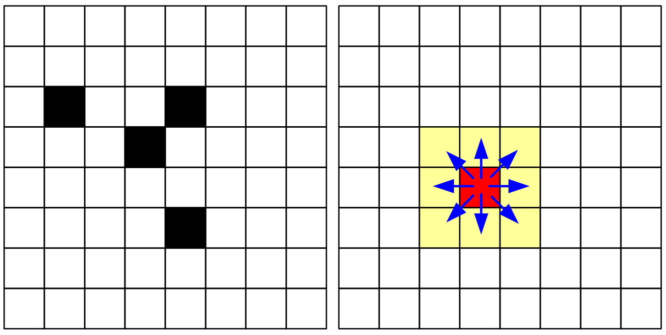

(a) Obstacles in the map **(b)** Walking direction of the robot

Fig. 1. Grid map and the movement directions of the robot

A* algorithm spreads all adjacent nodes so that the search time is long. To solve this problem, the node is restricted to moving in the directions of up, down, left, or right, and is not allowed to move diagonally in this paper. Therefore, each node in the grid map can be divided into the following five categories:

1. Quad-connected node: there are no obstacles in any of the four directions of movement at the current node, indicating that the node is free to move in any of the four directions, which are marked as N_4.
2. Tri-connected node: among the four directions of movement at the current node, only one direction is obstructed, allowing the node to move freely in any of the remaining three directions, which is marked as N_3.
3.) Bi-connected node: among the four directions of movement at the current node, two directions have obstacles, indicating that the node can freely move in any of the remaining two directions, which are marked as N_2.
4. Obstacle node: impassable node, marked as N_0.
5. Target node: the endpoint of the path is represented as N_T.

The nodes described above are shown in Fig. 2.

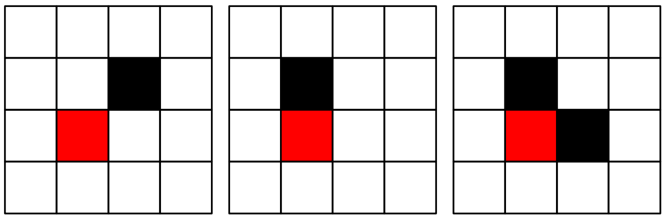

(a) Quad-connected node **(b)** Tri-connected node **(c)** Bi-connected node

Fig. 2. Node types of this paper

2.2. Critical node model description

The critical node in the robot global path planning is proposed in this paper, which is represented by N_{crit}. The critical node represents the mutation of the node type. If a node N in the grid map meets one of the following conditions, it is the critical node in this paper:

(1) When Quad-connected nodes are searching in a certain direction, the obstacle nodes or Tri-connected nodes appear in

front of them, which can be represented as follows:

$$N_{\text{crit}} = N\left(N_4 \xrightarrow{ANY} N_0|N_3\right). \tag{1}$$

Equation (1) represents that if the current node is Quad-connected, the critical node can be searched along any direction until a Tri-connected node is found or an obstacle is encountered in the search direction. The node found during this search is the critical node, denoted as N_{crit}.

(2) When the Quad-connected nodes are searching in a certain direction, the passing node is the same as the coordinate of the target node, which can be represented as follows:

$$N_{\text{crit}} = N\left(N_4 \xrightarrow{ANY} N_T(x)|N_T(y)\right). \tag{2}$$

Equation (2) represents that if the current node is Quad-connected, the critical node can be searched along any direction until the x-coordinate or y-coordinate is the same as the destination. The node found during this search is the critical node, denoted as N_{crit}.

(3) When a Tri-connected node or a Bi-connected node is searching along an obstacle, an obstacle appears in front of it, which can be represented as follows:

$$N_{\text{crit}} = N\left(N_2|N_3 \xrightarrow{OBS} N_0\right). \tag{3}$$

Equation (3) represents that if the current node is a Tri-connected node or a Bi-connected node, the critical node can be searched along the obstacle-free direction until an obstacle is encountered. The node found during this search is the critical node, denoted as N_{crit}.

(4) When a Tri-connected node or a Bi-connected node is searching along an obstacle, the passing node suddenly changes to a Quad-connected node, which can be represented as follows:

$$N_{\text{crit}} = N\left(N_2|N_3 \xrightarrow{OBS} N_4\right). \tag{4}$$

Equation (4) represents that if the current node is a Tri-connected node or a Bi-connected node, it is possible to search for critical nodes in the direction without obstacles until the x-coordinate or y-coordinate is the same as the endpoint. The nodes found during this search process are critical nodes, denoted as N_{crit}.

The four critical nodes described above can be shown in Figs. 3, 4, and 5.

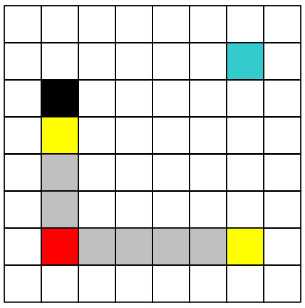


Fig. 3. Condition of meeting the Quad-connected critical node

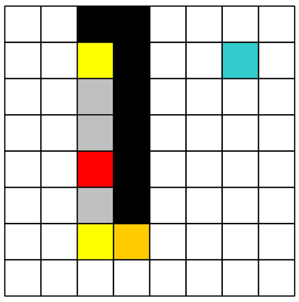


Fig. 4. Condition of meeting the Tri-connected critical node

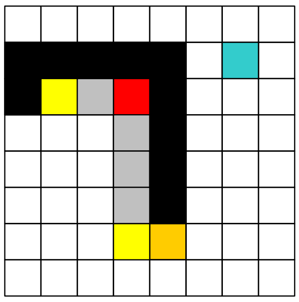


Fig. 5. Condition of meeting the Bi-connected critical node

In the figures above: the red node is the starting node, the blue node is the target node, the black node is the obstacle, the grey node is the path node in the search process, the yellow node is the critical node, the orange node is a special critical node, which will be added to the search list in diffusion.

Remark 1. When the critical node is found, the search mission in this direction is completed, and the found critical node is treated as the current node. Furthermore, the search direction will be confirmed by the critical node type.

Remark 2. To improve the search efficiency and applicability of the proposed algorithm, the current search direction is neglected in the next search process, which guarantees the search area is not repeated.

3. SEARCH ALGORITHM BASED ON CRITICAL-NODE DIFFUSION BINARY TREE

3.1. Critical-node diffusion binary tree

The critical node diffusion binary tree algorithm in this paper divides the finding process of the key nodes into two stages: the search stage and the diffusion stage. The search stage refers to finding a critical node in a certain direction from the previous critical node. The diffusion stage refers to specifying the multiple directions for the search task from the previous critical node. Both processes can be shown in Fig. 6.

In the figure above, the search stage is stopped until a critical node is found during the search process. All the critical nodes found in the search process are added to the OpenList, which is a priority queue that stores the critical nodes obtained during each search. In the OpenList, the critical nodes are arranged in ascending order based on the cost value of the critical nodes.

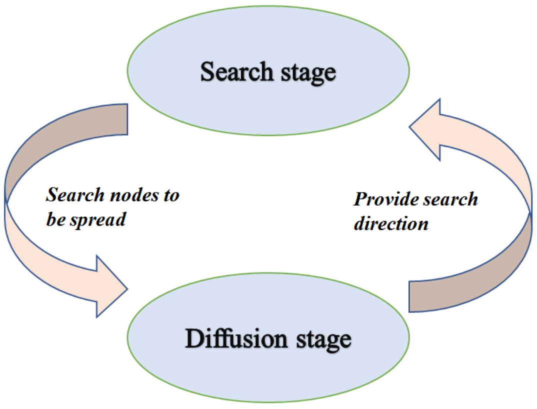


Fig. 6. Search and diffusion stages

Then the diffusion stage is started, in which the search directions are decided so that the search phase goes on again.

In the diffusion stage, if the start node is a Quad-connected one, the x-axis and y-axis of the node that points towards the target node are specified as the search directions. If the start node is a Tri-connected or Bi-connected one, the direction along the obstacle is specified as the searching direction. It is noted that the repeated search direction should be eliminated due to its parent node position.

In the search process, when the fourth type of a critical node (in Section 2.2) is found, these critical nodes and the nodes where the obstacle disappears are all added to the OpenList. The latter is a type of free critical node.

Remark 3. There is a strong purpose for the Quad-connected node. That means if the next critical node is the Quad-connected node (It is noted that this node is a four-connected node, which is neither the start node nor the target node), the target node will be finally searched. In this process, other critical nodes may be found, and result in repeated search missions. Therefore, when a Quad-connected node is searched, i.e., the node where the obstacle disappears is also regarded as a special critical node, which provides more effective search efficiency.

3.2. Critical-node diffusion binary tree algorithm

After the determination of the search direction and acquirement of the critical node in the diffusion and search stages, the cost function of all nodes stored in the OpenList is calculated as follows:

$$F_{\text{cost}} = G_{sc} + H_{ct}\,. \tag{5}$$

where, F_{cost} is the total generation value of the nodes, G_{sc} is the actual path from the start node to the current node. H_{ct} is the heuristic distance, which represents the estimated Euclidean distance from the current node to the target node. The node with the lowest generation value is selected as the current node to execute the subsequent diffusion and search stages. It is worth mentioning that the heuristic function only affects the shape of the path and the path-planning time. In other words, using any heuristic function, CDBT can find a path from the starting node to the target node.

The pseudo-code of the algorithm is as follows:

Algorithm 1. CDBT Algorithm

```
Input: $N_S$, $N_T$.
Output: A path $\Gamma$ from $N_S$ to $N_T$.
1:  $N_{\text{cur}} \leftarrow N_S$;
2:  $OpenList \leftarrow NULL$;
3:  $G_{N_S} \leftarrow 0$;
4:  while ($N_{\text{cur}} \neq N_T$) do
5:      if ($N_{cur} = N_4$) then
6:          $N_{\text{crit}} \leftarrow N(N_4 \xrightarrow{ANY} N_0|N_3|N_T(x)|N_T(y))$;
7:  end if
8:      if ($N_{\text{cur}} = N_3|N_2$) then
9:          $N_{\text{crit}} = N(N_2|N_3 \xrightarrow{OBS} N_0|N_4)$;
10: end if
11:     $N_{\text{crit}} \xrightarrow{father} N_{\text{cur}}$;
12:     $G_{N_{\text{crit}}} \leftarrow G_{N_{\text{cur}}} + D_{N_{\text{cur}} \to N_{\text{crit}}}$;
13:     $F_{N_{\text{crit}}} \leftarrow G_{N_{\text{crit}}} + H_{N_{\text{crit}} \to N_T}$;
14:     $OpenList \xleftarrow{push\_back} N_{\text{crit}}$;
15: for ($N_{\text{crit}}$ in $OpenList$) do
16:         $N_{\text{cur}} \leftarrow N_{F_{\min}}$;
17: end for
18: end while
19:     $N_{\text{cur}} \leftarrow N_T$;
20: while ($N_{\text{cur}} \neq NULL$) do
21:     $\Gamma \xleftarrow{push\_back} N_{\text{cur}}$;
22:     $N_{\text{cur}} \leftarrow N_{f_{\text{cur}}}$;
23: end while
24: return $\Gamma$;
```

In the table above, $OpenList$ is the set of expansion nodes, G_*, H_*, and F_* are the real cost, estimated cost, and total cost of a node $*$, respectively. f_* is the parent node of the node $*$.

The start position of the robot is the current node, and the destination is the target node. G_{sc} of the start, node is set as 0. If the current node is Quad connected, the target node is searched along the x-axis and y-axis until the critical node is acquired. If the current node is Tri-connected, the parent node of the current node is recorded, and then the direction of the parent node will not be searched during searching along the obstacle. When a special node (mentioned by Remark 3) is encountered in the search process, the critical node and this special node are also marked as the critical nodes. All the critical nodes are added to the OpenList and then are calculated, and the current node is selected as the parent node of the critical nodes that are added to the OpenList during this search stage. The node with the minimal F_{cost} in the OpenList is selected as the current node, and the search and diffusion stages are repeated until the current node is the destination.

Remark 4. The search strategy of A* is to calculate all the neighbour nodes around the current node. In other words, A* has to judge eight nodes (around the current node) in each search process, even if these nodes were judged in the previous judgment. On the contrary, the CDBT in our manuscript only judges one, two, or three nodes. Though the computational complexity of CDBT and A* cannot be described by a specific expression, the computational complexity of CDBT is lower than that of A* in all kinds of scenarios from the perspective of node search strategy.

4. SECONDARY OPTIMIZATION BASED ON REVERSE SEARCHING AND ROBOT CONSTRAINTS

4.1. Reverse parent node selection

In global path planning, it is generally desired to minimize the number of turning points on the desired path, to achieve a smoother trajectory. Additionally, it is important to ensure that the direction of the path does not change abruptly, as this promotes stability in the robot movement and enhances its overall efficiency. In this article, the smoothness of the path refers to the number of turning points in the path. The fewer inflection points, the higher the smoothness of the path. Motion efficiency, on the other hand, refers to the angle of change in direction when the path changes. The smaller the angle, the higher the motion efficiency is considered to be. There are redundant nodes in the feasible path in Section 3. Thus, a reverse searching method is designed to minimize the nodes, which guarantees the smoothness of the motion path and improves the motion efficiency of the robot.

In the initial feasible path, a trace backward is conducted from the target node to the start node. The n_ith and n_{i-2}th nodes can be connected and the n_{i-1} can be eliminated under the condition of the following equation:

$$\zeta(n_i, n_{i-2}) = \emptyset, \quad n_{i-2} \neq 0, \quad i = N, N-1, \ldots, 0, \tag{6}$$

where, n_i and n_{i-2} are the two nodes selected in the reverse tracing process. $\zeta(*,*)$ represents whether the connection between both nodes passes through the obstacle. $\zeta(*,*) = \emptyset$ indicates that it does not pass through the obstacle. Otherwise, it indicates that it passes through the obstacle. The equation above completes the path-smoothing process.

The pseudo-code for path optimization is as follows:

Algorithm 2. Reverse node finding algorithm

```
Input: the old path $\Gamma$
Output: a new path P
1:  $N_{cur} \xleftarrow{tail\_node} \Gamma$;
2:  while ($f_{N_{cur}} \neq NULL$) do
3:      $N_1 \leftarrow f_{f_{N_{cur}}}$;
4:      if (No obstacle between $N_{cur}$ and $N_1$) then
5:          $f_{N_{cur}} \leftarrow f_{f_{N_{cur}}}$;
6:      Else
7:          $N_{cur} \leftarrow f_{N_{cur}}$;
8:      end if
9:  end while
10: $N_{cur} \xleftarrow{tail\_node} \Gamma$;
11: while ($N_{cur} \neq NULL$) do
12:     $P \xleftarrow{push\_back} N_{cur}$;
13:     $N_{cur} \leftarrow N_{f_{cur}}$;
14: end while
15: return P;
```

The flow of the critical node diffusion-based binary tree search algorithm and the path optimization algorithm based on reverse searching are shown in Fig. 7.

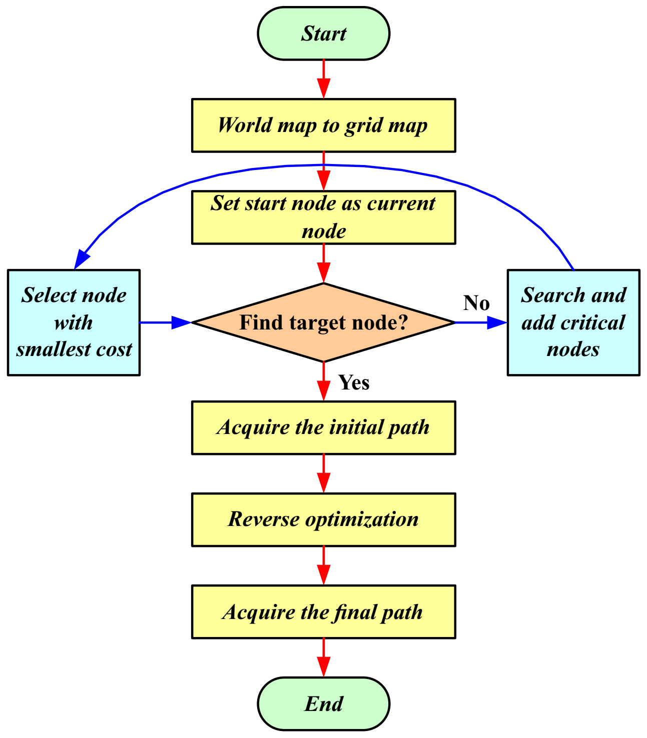
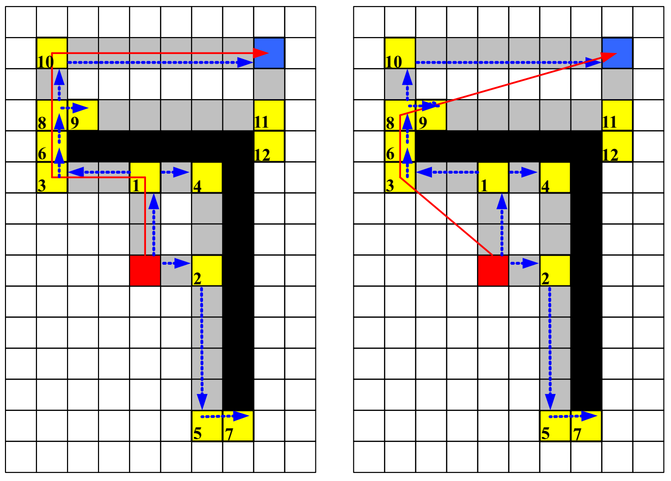


Fig. 7. Reverse search schematic diagram

Taking the grid map in Fig. 8a as an example, the planning and optimization effects of the critical node diffusion-based binary tree search algorithm and the path optimization algorithm based on reverse searching are as follows in Fig. 8b.



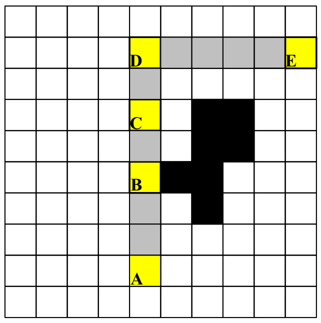
(a) Path before optimization **(b)** Path after optimization

Fig. 8. Path reverse optimization

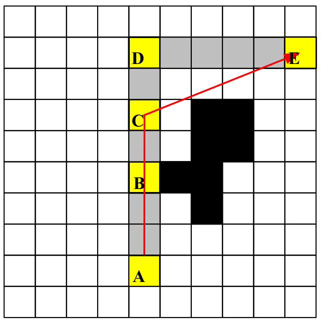
Figures 8a and 8b illustrate that the optimization algorithm eliminates unnecessary path-turning nodes and can quickly find paths with fewer turning nodes from the starting node to the target node, improving the overall smoothness of the path.

4.2. Path secondary search based on critical nodes

The planned path in Section 4.1 does not consider the maximum steering angle of the actual robot. A secondary search scheme based on critical nodes is proposed, which optimizes the global path twice. The principle of this method is to make the path corner so small that the burden of subsequent local planning is reduced. The principle can be shown in Fig. 9.



(a) Before the first optimization



(b) After the first optimization

Fig. 9. Path optimization for the first time

In the figure above, yellow nodes are critical ones, black nodes are obstacles, and grey nodes are the search direction. When there is no secondary search, the planned path is shown in Fig. 9b. The path corner at the critical node C is large, which may be larger than the maximum steering angle of the robot. The path shown in Fig. 9b is searched again. The path corner of the planned route at the processed critical node C is as follows:

$$\theta_C = \arccos\left(\frac{\overrightarrow{EC}\cdot\overrightarrow{CA}}{\left\|\overrightarrow{EC}\right\|\cdot\left\|\overrightarrow{CA}\right\|}\right) \geq \delta, \tag{7}$$

where, $\|*\|$ represents the Euclidean distance. δ is the maximum steering angle of the robot. It is important to note that some types of planar mobile robots such as differential drive robots and omnidirectional robots can rotate in place without requiring a minimum turning radius. However, to achieve a smoother trajectory for the entire robot, it is necessary to utilize the optimization methods mentioned earlier. In such cases, the minimum turning radius of the vehicle can be set as either the radius of the robot or the length of the vehicle.

In Fig. 10, a new critical node O on the segment AB is searched, where A and B are the previous two critical nodes of the processed critical node C.

Therefore, the following equation is satisfied:

$$\begin{cases} \hat{\theta}_C \leq \delta, \\ \hat{\theta}_O \leq \delta, \end{cases} \tag{8}$$

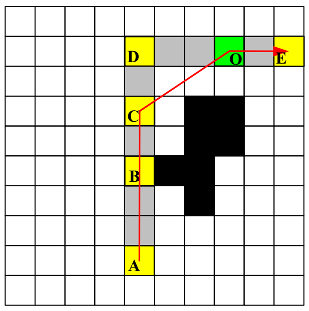


Fig. 10. Secondary path optimization

where, $\hat{\theta}_C$ and $\hat{\theta}_O$ are shown below:

$$\begin{cases} \hat{\theta}_C = \arccos\left(\frac{\overrightarrow{AC}\cdot\overrightarrow{CO}}{\left\|\overrightarrow{AC}\right\|\cdot\left\|\overrightarrow{CO}\right\|}\right), \\ \hat{\theta}_O = \arccos\left(\frac{\overrightarrow{CO}\cdot\overrightarrow{OE}}{\left\|\overrightarrow{CO}\right\|\cdot\left\|\overrightarrow{OE}\right\|}\right). \end{cases} \tag{9}$$

The path corner at each critical node can be less than the maximum steering angle of the robot as far as possible, which makes the global path consistent with the robot motion.

5. EXPERIMENT AND ANALYSIS

In this section, we conducted a total of four experiments to compare four algorithms: CDBT, A*, RRT, and RRT*. We used path length, planning time, number of expanded nodes, and number of turning points as indicators. The first experiment displayed the expanded nodes for the four algorithms. The second and third experiments compared the four algorithms on the same map and different maps, respectively. The final experiment applied the CDBT algorithm to a real-world scenario.

5.1. Comparison of the number of expansion nodes

Firstly, we specified the starting and ending points on a blank map, and the diffusion nodes corresponding to the four path-planning algorithms are shown in Fig. 11.

As shown in Fig. 11 and Table 1, the green nodes depict the paths planned by different algorithms, and the red nodes signify the expansion nodes required by each algorithm. It is evident from the figure that the A* algorithm exhibits a higher

Table 1
Expansion nodes number

Algorithm	The number of expansion nodes
A*	168
CDBT	12
RRT	48
RRT*	1458

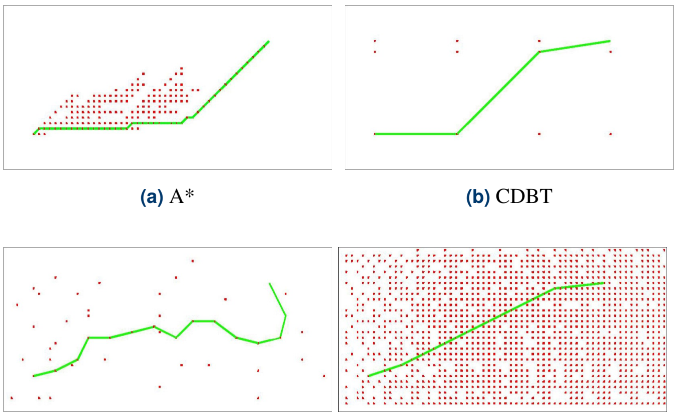

(a) A* (b) CDBT (c) RRT (d) RRT*

Fig. 11. Comparison of the number of expansion nodes

number of expansion nodes due to its selection of only adjacent nodes for diffusion. The RRT algorithm obtains expansion nodes through random sampling, resulting in a relatively smaller number of nodes compared to A*. The RRT* algorithm necessitates continuous sampling for path optimization once it is obtained, leading to the highest number of expansion nodes among the four algorithms. Fortunately, the path generated by RRT* is considerably smoother than that of RRT. The CDBT algorithm selectively chooses critical nodes for diffusion, resulting in the fewest expansion nodes compared to the aforementioned three algorithms. This is also the primary factor enabling the CDBT algorithm to plan faster than the other three algorithms.

5.2. Comparison on the same map

Firstly, we conducted experiments on the same map, selecting four different starting points and destinations for comparison of the four indicators mentioned above, as shown in Fig. 12.

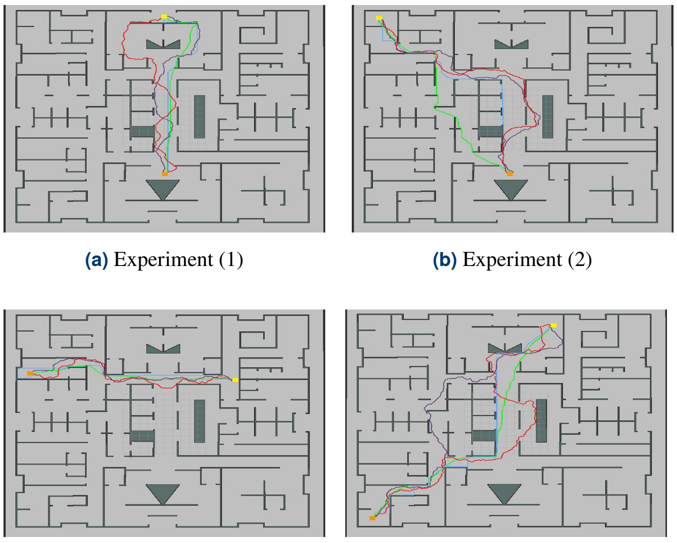

(a) Experiment (1) (b) Experiment (2) (c) Experiment (3) (d) Experiment (4)

Fig. 12. Comparison of algorithms on the same map (The paths planned by the A*, RRT, RRT*and CDBT algorithms are depicted by the green, red, purple, and blue paths, respectively.)

The yellow nodes in Fig. 12 indicate the starting point, while the orange nodes represent the destination. Table 2 and Figs. 13–16 illustrate the comparison of these four algorithms based on four indicators.

Table 2
Comparison of algorithms on the same map

Method	ID	Path length (m)	Expansion nodes number	Path-turning nodes number	Path-planning time (ms)
A*	(1)	31	20926	82	805.4
A*	(2)	35	44723	105	1535.8
A*	(3)	26	12142	67	898.7
A*	(4)	44	76856	161	2512.3
CDBT	(1)	33	1004	13	9.3
CDBT	(2)	38	3211	13	63.1
CDBT	(3)	31	278	11	3.2
CDBT	(4)	48	8792	13	438.6
RRT	(1)	39	4679	75	245.1
RRT	(2)	45	2252	65	56.1
RRT	(3)	34	394	82	4.1
RRT	(4)	63	6029	165	343.4
RRT*	(1)	36	19832	70	1690.6
RRT*	(2)	44	24478	79	2392.3
RRT*	(3)	30	25114	43	1611.2
RRT*	(4)	55	32786	96	2824.1

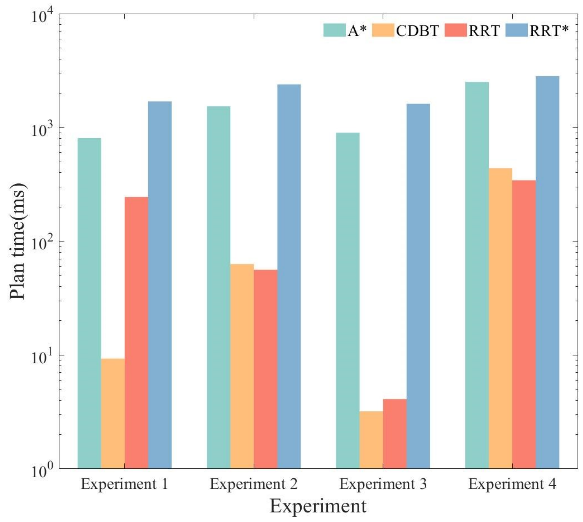


Fig. 13. Comparison of path-planning time on the same map

As shown in Figs. 12–15, it can be observed that both CDBT and RRT achieve paths in a similar amount of time for path planning, which is faster than A* and RRT*. In terms of path length, the path length obtained by the CDBT algorithm is approximate

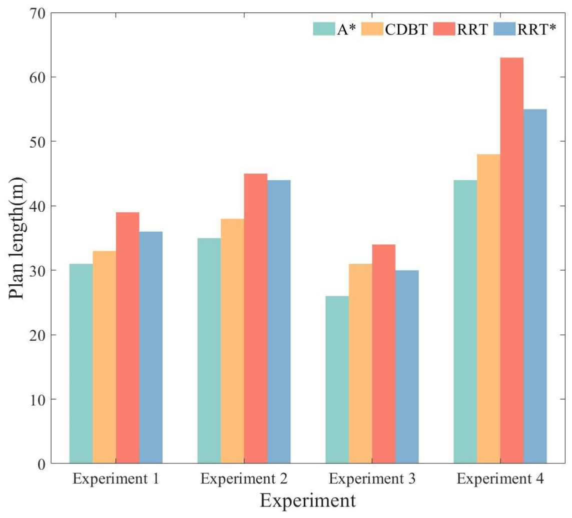


Fig. 14. Comparison of path length on the same map

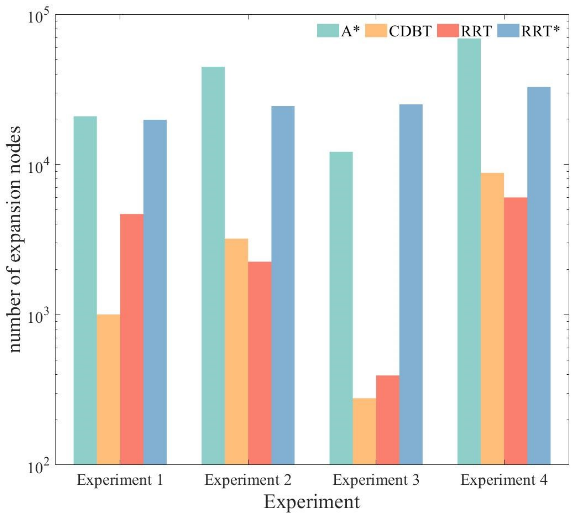


Fig. 15. Comparison of expansion nodes on the same map

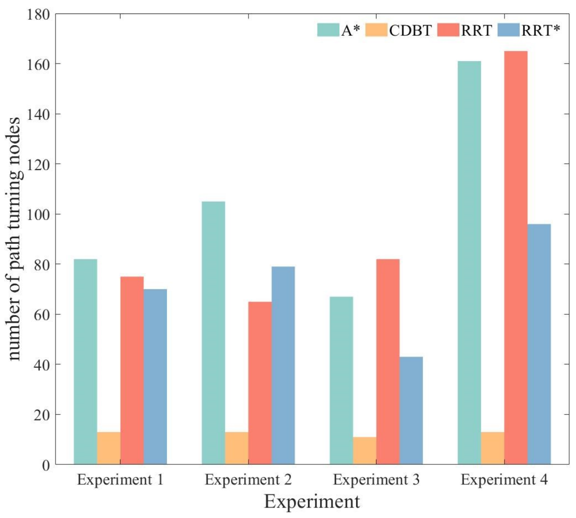


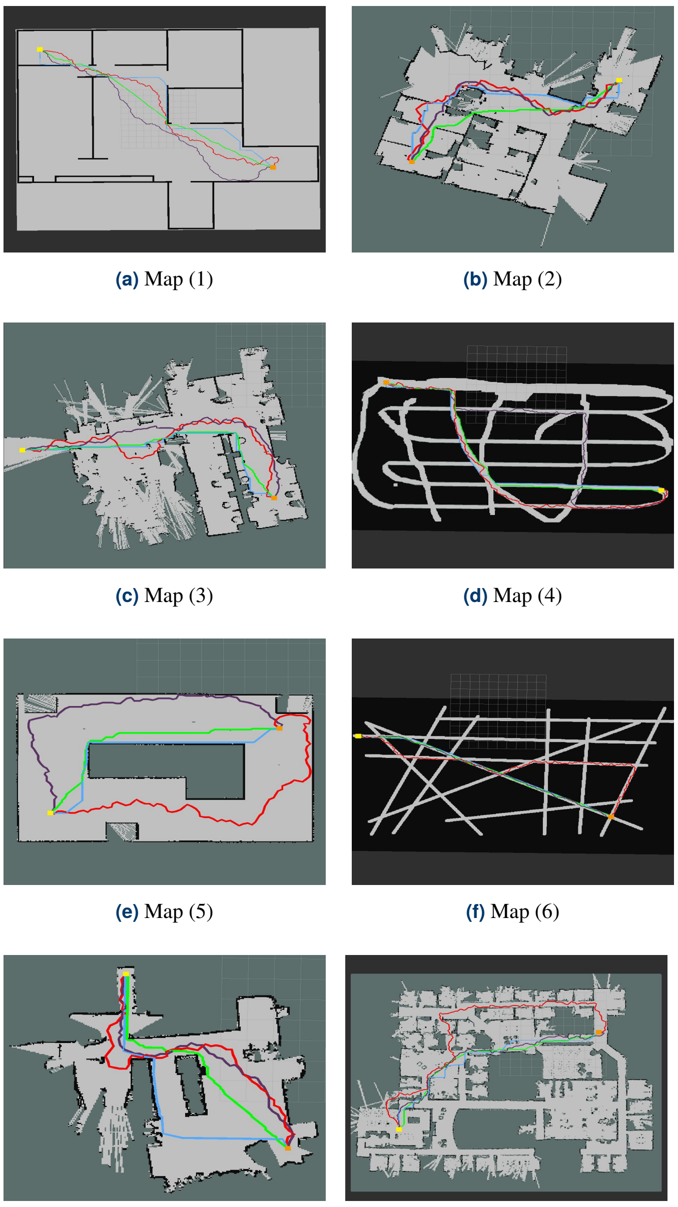
Fig. 16. Comparison of turning nodes on the same map

to the path length obtained by A*. CDBT also outperforms in terms of the number of expanded nodes and turning points. These findings suggest that CDBT possesses the capability to rapidly plan paths similar to RRT while maintaining path lengths similar to A*.

To further validate the superiority of the CDBT algorithm, we conducted experiments on eight diverse types of maps.

5.3. Comparison of different maps

To further validate the path-planning performance of CDBT, a comparison was conducted between A*, RRT, RRT*, and CDBT in eight different simulated environments. The map types included simple maps, complex scenes, and large-scale maps. The comparison results are shown in Fig. 17.



(a) Map (1) **(b)** Map (2) **(c)** Map (3) **(d)** Map (4) **(e)** Map (5) **(f)** Map (6) **(g)** Map (7) **(h)** Map (8)

Fig. 17. Comparison of algorithms on different maps (The green, red, purple, and blue paths in the figure represent the planned routes by the A*, RRT, RRT*, and CDBT algorithms, respectively.)

The yellow nodes in Fig. 17 denote the initial point, while the orange nodes indicate the target location. Table 3 and Figs. 18–21 present a comparison of these four algorithms using four metrics.

Table 3
Comparison of algorithms on eight maps

Method	Map ID	Path length (m)	Expansion nodes number	Path-turning nodes number	Path-planning time (ms)
A*	(1)	39	46275	149	1165.7
A*	(2)	12	5316	44	107.8
A*	(3)	21	13883	77	405.0
A*	(4)	37	33465	132	808.3
A*	(5)	14	11371	35	683.8
A*	(6)	31	6243	183	154.1
A*	(7)	12	4012	36	56.5
A*	(8)	26	19620	81	561.9
CDBT	(1)	46	5451	12	110.9
CDBT	(2)	15	2084	19	17.1
CDBT	(3)	22	6350	22	143.9
CDBT	(4)	38	4419	44	250.3
CDBT	(5)	14	427	5	2.3
CDBT	(6)	30	5061	13	100.7
CDBT	(7)	13	2364	6	22.5
CDBT	(8)	28	7879	25	341.3
RRT	(1)	48	5141	97	157.5
RRT	(2)	17	804	37	15.0
RRT	(3)	27	960	62	9.8
RRT	(4)	48	1167	83	20.9
RRT	(5)	21	1437	54	32.8
RRT	(6)	43	1650	103	47.5
RRT	(7)	16	387	32	10.9
RRT	(8)	42	2213	85	79.7
RRT*	(1)	46	25028	87	2738.2
RRT*	(2)	16	4032	40	387.7
RRT*	(3)	24	10810	56	1523.8
RRT*	(4)	45	7501	75	1029.5
RRT*	(5)	19	5284	48	550.5
RRT*	(6)	31	4231	67	431.6
RRT*	(7)	14	2298	23	200.7
RRT*	(8)	29	14001	70	2242.2

In the figure above, four path-planning methods can achieve global path planning. However, the CDBT method has significantly fewer expansion nodes compared to other planning methods. Based on previous experiments, it can be demonstrated that

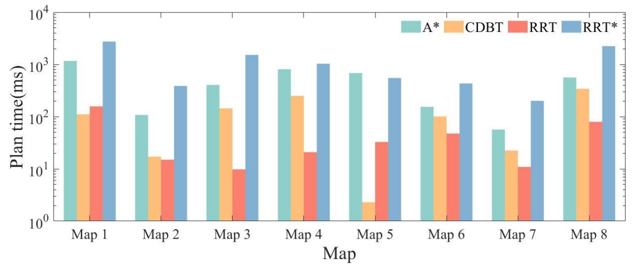


Fig. 18. Comparison of path-planning time on different maps

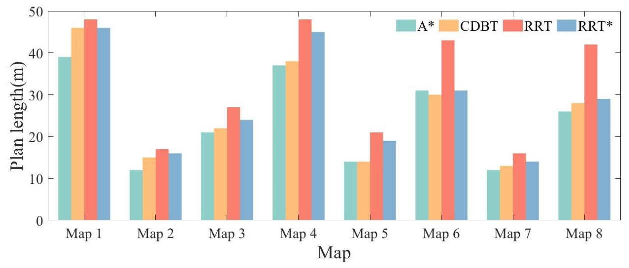


Fig. 19. Comparison of path length on different maps

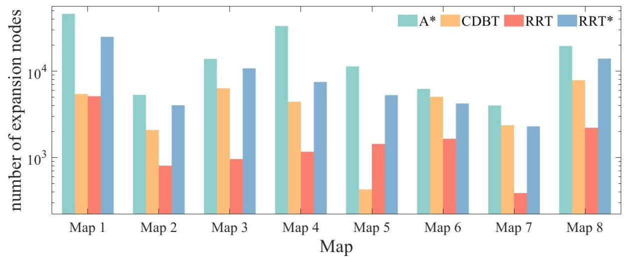


Fig. 20. Comparison of expansion nodes on different maps

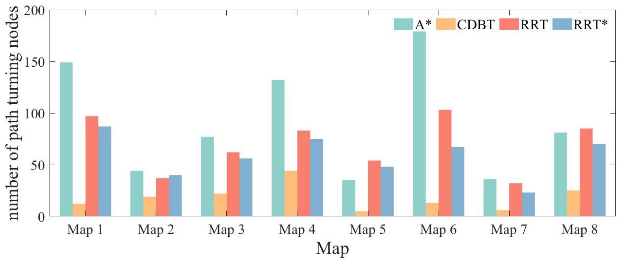


Fig. 21. Comparison of path-turning nodes on different maps

the CDBT algorithm possesses fast path-planning capabilities similar to RRT while maintaining path lengths similar to A*. In summary, the CDBT algorithm can quickly plan a path with a length close to that planned by the A* algorithm, with much fewer turning points than the A* path. This is highly beneficial for robot motion, as it greatly reduces the challenges associated with robot control and subsequent path smoothing.

5.4. Experiment with robot navigation

To verify the practicability of the actual robots, the Qingzhou robot is selected to complete an experiment based on the proposed CDBT method, which can be seen in Fig. 22. The Qingzhou robot is an unmanned learning robot, which provides a set of hardware platforms. The Ackerman structure in the Qingzhou robot provides the maximum steering radius and angle of 35 cm and 45 degrees, respectively. The LiDAR on the robot is Ydlidar X4 with a maximum measurement distance of 10 m. The mapping method for the robot adopted the Gmapping algorithm, which is not described here.

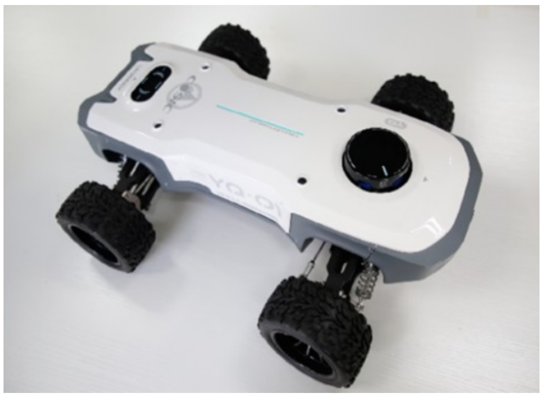


Fig. 22. Qingzhou robot

Like A* and RRT, CDBT is a global path-planning algorithm that can only be used in static maps. The path generated by CDBT only includes position information and does not contain trajectory information such as velocity and acceleration. The global path serves as a rough planning direction for local path planning. CDBT can be combined with existing local path-planning algorithms to control vehicle motion.

After determining the starting and ending points of the robot, the CDBT algorithm is employed to plan the global path. Once the robot acquires the path information, it utilizes the DWA algorithm for trajectory tracking. However, due to the limited length of this article and the fact that the DWA algorithm is not the primary focus of this research, this article only employs DWA to verify the feasibility of CDBT path planning. The principles of DWA will not be further elaborated on.

The CDBT algorithm is deployed to the Qingzhou robot. The experiment scene is an indoor environment, which is shown in Fig. 23.

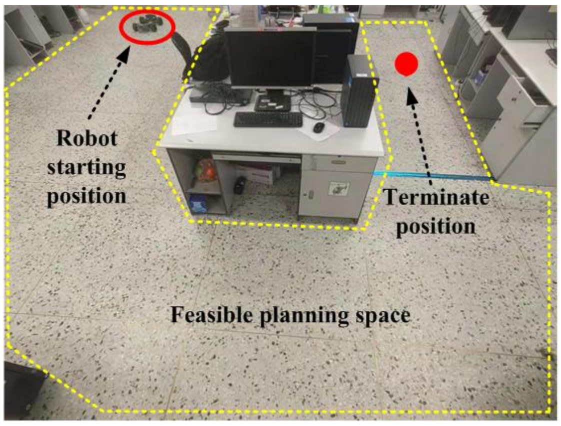


Fig. 23. Experiment environment

The planning and actual paths of the robot are shown in Fig. 24.

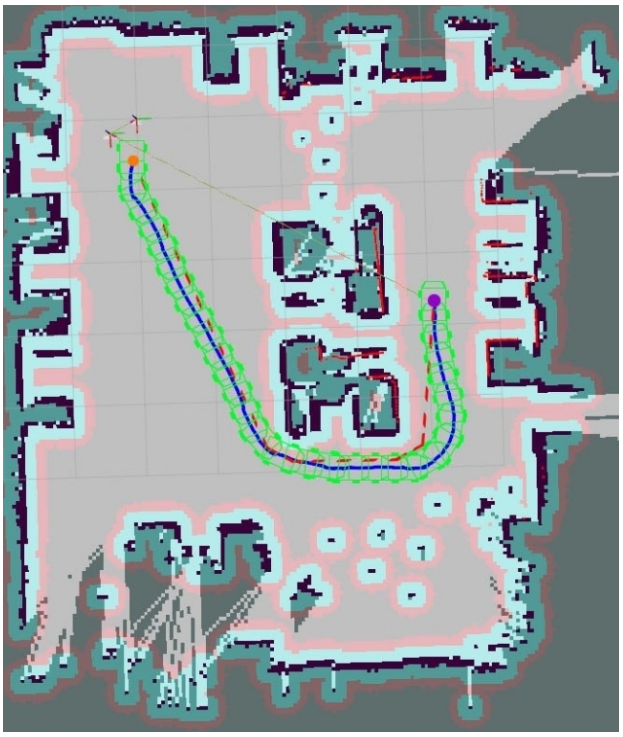


Fig. 24. Planning and actual paths of robot

In Fig. 24, the orange point is the start point, the green point is the goal point, the blue dotted line is the global path, and the red solid line is the actual driving path of the robot. It can be seen that the CDBT algorithm can plan the path of the robot, and the final path of the Qingzhou robot is highly consistent with global planning.

6. CONCLUSION

A robot global path-planning algorithm is developed, which is based on the diffusion binary tree of critical nodes. This algorithm effectively addresses the issues of low efficiency, excessive path-turning nodes, and lengthy traditional path optimization. The following conclusions are obtained: (1) the proposed algorithm can improve the path optimization efficiency by diffusing the critical node and improve the path smoothness by combining the path reverse optimization process so that the robot can reach the target node efficiently and safely. (2) In this paper, the CDBT algorithm is second only to A* in path length and approximates RRT in planning time. It also has superior performance in terms of the number of path expansion nodes and path turning nodes. (3) An experiment with the Qingzhou robot is completed, which verifies the effectiveness of the proposed method. In conclusion, the new global path-planning algorithm proposed in this paper has a beneficial effect on path-turning nodes, optimization time, etc. However, global path planning does not consider local path planning in the presence of dynamic obstacles, which will be further studied in the future.

ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (62173103), and by the Fundamental Research Funds for the Central Universities of China (No. 3072022JC0402, 3072022JC0403).

NOMENCLATURE

$\xrightarrow{*}$	the search in a certain direction, and $*$ on the arrow can be ANY or OBS, which means searching in any direction or along the edge of the obstacle, respectively
N_2, N_3, N_4	Bi-connected node, Tri-connected node, and Quad-connected node
N_0	the obstacle node
N_S	the start node. $N_S(x)$ and $N_S(y)$ are the coordinates of the start node
N_T	the target node. $N_T(x)$ and $N_T(y)$ are x and y coordinates of the target node

REFERENCES

[1] W. Kowalczyk and K. Kozlowski, “Trajectory tracking and collision avoidance for the formation of two-wheeled mobile robots,” *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 67, no. 5, pp. 915–924, 2019.

[2] U. Libal and J. Plaskonka, “Noise sensitivity of selected kinematic path following controllers for a unicycle,” *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 62, no. 1, pp. 3–13, 2014, doi: 10.2478/bpasts-2014-0001.

[3] W. Chi, Z. Ding, and J. Wang, “A Generalized Voronoi Diagram-Based Efficient Heuristic Path Planning Method for RRTs in Mobile Robots,” *IEEE Trans. Ind. Electron.*, vol. 69, no. 5, pp. 4926–4937, 2021, doi: 10.1109/TIE.2021.3078390.

[4] J. Yang, C. Wang, and B. Jiang, “Visual perception enabled industry intelligence: state of the art, challenges and prospects,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 3, pp. 2204–2219, 2020, doi: 10.1109/TII.2020.2998818.

[5] K. Shu, H. Yu, and X. Chen, “Autonomous driving at intersections: A behavior-oriented critical-turning-point approach for decision making,” *IEEE-ASME Trans. Mechatron.*, vol. 27, no. 1, pp. 234–244, 2021, doi: 10.1109/TMECH.2021.3061772.

[6] K. Wu, H. Wang, and M. A. Esfahani, “Achieving real-time path planning in unknown environments through deep neural networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2093–2102, 2020, doi: 10.1109/TITS.2020.3031962.

[7] Q. Wei, H. Li, and X.S. Yang, “Continuous-time distributed policy iteration for multicontroller nonlinear systems,” *IEEE T. Cybern.*, vol. 51, no. 5, pp. 2372–2383, 2020, doi: 10.1109/TCYB.2020.2979614.

[8] S. Zhao, L. Shi, and W. Zhang, “Global dynamic path-planning algorithm in gravity-aided inertial navigation system,” *IET Signal Process.*, vol. 15, no. 80, pp. 510–520, 2021, doi: 10.1049/sil2.12056.

[9] V. Venkatesan, J. Seymour, and D. J.Cappelleri, “Micro-assembly sequence and path planning using subassemblies,” *J. Mech. Robot.*, vol. 10, no. 6, 2018, doi: 10.1115/1.4041333.

[10] E. Çakır, Z. Ulukan, and T. Acarman, “Time-dependent Dijkstra’s algorithm under bipolar neutrosophic fuzzy environment,” *J. Intell. Fuzzy Syst.*, vol. 42, no. 1, pp. 227–236, 2022, doi: 10.3233/JIFS-219188.

[11] G. Dong, F. Yang, and K. L. Tsui, “Active Balancing of Lithium-Ion Batteries Using Graph Theory and A-Star Search Algorithm,” *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2587–2599, 2021, doi: 10.1109/TII.2020.2997828.

[12] P. Skačkauskas and E. Sokolovskij, “Analysis of the Hybrid Global Path Planning Algorithm for Different Environments,” *Transp. Telecommun. J.*, vol. 20, no. 1, pp. 1–11, 2019, doi: 10.2478/ttj-2019-0001.

[13] S. Kadry, G. Alferov, and V. Fedorov, “D-Star Algorithm Modification,” *Int. J. Online Biomed. Eng.*, vol. 16, no. 8, pp. 108–113, 2020, doi: 10.3991/ijoe.v16i08.14243.

[14] N. Ma, J. Wang, and J. Liu, “Conditional Generative Adversarial Networks for Optimal Path Planning,” *IEEE Trans. Cogn. Dev. Syst.*, vol. 14, no. 2, pp. 662–671, 2022, doi: 10.1109/TCDS.2021.3063273.

[15] X. Song, S. Gao, and C. B. Chen, “A New Hybrid Method in Global Dynamic Path Planning of Mobile Robot,” *Int. J. Comput. Commun. Control*, vol. 13, no. 6, pp. 1032–1046, 2022, doi: 10.15837/ijccc.2018.6.3153.

[16] Y. Long, Z. Zuo, and Y. Su, “An A*-based bacterial foraging optimisation algorithm for global path planning of unmanned surface vehicles,” *J. Navig.*, vol. 73, no. 6, pp. 1–16, 2020, doi: 10.1017/S0373463320000247.

[17] I.S. Kim, W.K. Lee, and Y.D. Hong, “Simple global path planning algorithm using a ray-casting and tracking method,” *J. Intell. Robot. Syst.*, vol. 90, no. 6, pp. 101–111, 2018, doi 10.1007/s10846-017-0642-2.

[18] K. Zhang, Y. Yang, andM. Fu, “Two-phase A*: A real-time global motion planning method for non-holonomic unmanned ground vehicles,” *Proc. Inst. Mech. Eng. Part D-J. Automob. Eng.*, vol. 235, no. 4, pp. 1–16, 2021, doi: 10.1177/0954407020948397.

[19] J. Zhang, Y. Xia, and G. Shen, “A novel learning-based global path planning algorithm for planetary rovers,” *Neurocomputing*, vol. 361, no. 1, pp. 69–76, 2019, doi: 10.1016/j.neucom. 2019.05.075.

[20] G. Xia, Z. Han, and B. Zhao, “Global path planning for unmanned surface vehicle based on improved quantum ant colony algorithm,” *Math. Probl. Eng.*, vol. 7, pp. 1-10, 2019, doi: 10.1155/2019/2902170.

[21] C. Huang and J. Fei, “UAV path planning based on particle swarm optimization with global best path competition,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 32, no. 1, pp. 1–23, 2018, doi: 10.1142/S0218001418590085.

[22] P.G. Luan, and N.T. Thinh, “Hybrid genetic algorithm based smooth global-path planning for a mobile robot,” *Mech. Based Des. Struct. Mech.*, vol. 51, no. 3, pp. 1758–1774, 2023, doi: 10.1080/15397734.2021.1876569.

[23] Z. Yu and L. Xiang, “NPQ-RRT: an improved RRT approach to hybrid path planning,” *Complexity*, vol. 2021, p. 6633878, 2021, doi: 10.1155/2021/6633878.

[24] Q. Song, S. Li, and J. Yang, “Intelligent Optimization Algorithm-Based Path Planning for a Mobile Robot,” *Comput. Intell. Neurosci.*, vol. 2021, p. 8025730, 2021, doi: 10.1155/2021/8025730.

[25] J. Wang, Y. Luo, and X. Tan, “Path Planning for Automatic Guided Vehicles (AGVs) Fusing MH-RRT with Improved TEB,” *Actuators*, vol. 10, no. 12, p. 314, 2021, doi: 10.3390/act10120314.

[26] D. Pazderski, ”Application of transverse functions to control differentially driven wheeled robots using velocity fields,” *Bull. Pol. Acad. Sci. Tech. Sci.*, vol. 64, no. 4, pp. 831–851, 2016, doi: 10.1515/bpasts-2016-0092.