

Received 6 November 2022, accepted 25 November 2022, date of publication 2 December 2022, date of current version 8 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3226465

RESEARCH ARTICLE

CAF-RRT*: A 2D Path Planning Algorithm Based on Circular Arc Fillet Method

BINPENG WANG^{1,2}, DIANYUAN JU^{1,2}, FANGZHOU XU^{2,3}, (Member, IEEE),

CHAO FENG^{2,3}, AND GUANGLONG XUN⁴

¹School of Information and Automation, Shandong Academy of Sciences, Qilu University of Technology, Jinan 250353, China

²Jinan Engineering Laboratory of Human-Machine Intelligent Cooperation, Jinan 250353, China

³International School for Optoelectronic Engineering, Shandong Academy of Sciences, Qilu University of Technology, Jinan 250353, China

⁴Shandong MAIYUE Wenbao Technology Company Ltd., Jinan 250117, China

Corresponding author: Chao Feng (cfeng@qlu.edu.cn)

This work was supported by the Program for Youth Innovative Research Team in the University of Shandong Province in China (No. 2019KJN010).

ABSTRACT To improve the previous versions of Rapidly-exploring Random Trees (RRT) algorithms, a novel algorithm based on Circular Arc Fillet (CAF-RRT*) for path planning problems in two-dimension workspaces is proposed. Firstly, it is implemented by combining Quick-RRT* with the bidirectional RRT algorithm, which can obtain an initially planned path. Secondly, the cost of the initial planned path cost is reduced by a proposed path optimization strategy based on the triangle rule. Finally, based on the circular arc fillet method, a path smoothing strategy is proposed to process the middle nodes of the path, while further reducing its path cost. The experimental results on the MATLAB simulation platform show that compared with other RRT algorithms, the proposed algorithm can reduce the path length by about 2%, ensure the smoothness and security of the path, while increasing the running speed of the algorithm by about 95%, and the experimental results of the robot based on Robot Operating System (ROS) platform also prove the advantages of the algorithm.


INDEX TERMS CAF-RRT*, mobile robot, path planning, path smoothing, RRT.

I. INTRODUCTION

Path planning of mobile robots is one of the basic problems of the robotics research field [1]. It includes but is not limited to, robot navigation, self-driving cars, unmanned aerial vehicles, and so on [2]. This problem can be solved by a grid-based search algorithm or a sampling-based algorithm [3]. Grid-based search algorithms, such as A* [4], and sampling-based algorithms, such as Rapid Search Random Tree (RRT) [5] and Probabilistic Roadmap (PRM) [6], need to establish a set of trajectories through repeated random sampling. Theoretically, if the solution is found through infinite iterations, the probability can reach 100%.

To plan the path from the initial position to the target position, RRT needs to randomly sample in the search space and obtain its edge [7]. The disadvantage is that the path cost is high. Kuffner et al. [8] proposed RRT-Connect, which can

generate two spanning trees at starting position and target position at the same time, and the two trees simultaneously and alternately expand incrementally towards each other until the two trees are connected. However, RRT-Connect lacks the optimization process, and its path cost and convergence speed are not optimal. Wu et al. [9] proposed Fast-RRT, which only sampled in unexplored space of random tree through a fast sampling strategy, which improved the search speed and the stability of the algorithm, and proposed a random steering strategy to solve the problem of poor performance of the algorithm in narrow channels. By merging and adjusting paths, Fast-RRT can find the near-optimal path faster. However, Fast-RRT does not consider the real volume of the mobile robot in practical application, nor does it add the related algorithm of obstacle expansion, so the safety of the mobile robot cannot be guaranteed and the path is not smooth enough. Kang and Jung et al. [10] proposed Shorter-RRT, which can overcome the optimization limitation of the sampling-based algorithm. By deleting redundant intermediate nodes without

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

conflict. The proposed post-triangle rewiring method creates a path closer to the optimal one than RRT through the triangle inequality principle. However, the optimization of Shorter-RRT depends on the initial path obtained by RRT, so the optimization effect is limited, the optimal or suboptimal path cannot be guaranteed, and the security of the path in practical application is not considered. Mashayekhi et al. [11] proposed Hybrid-RRT, which first found the initial solution through bidirectional search, merged two trees into one tree, and then restricted the sampling area to an ellipse to optimize the current solution. The algorithm was simulated in the Open Motion Planning Library (OMPL). However, the restricted sampling area is easily affected by map factors, especially the complex maze map may not perform well.

Karaman and Frazzoli proposed RRT* [12], which can quickly calculate the initial path, and then optimize the path with the increase of sampling points until the target point is found or the set maximum number of iterations is reached. RRT* improves the parent node selection method, first selects the node with the lowest cost in the neighborhood of the node as the parent node, then reconnects the nodes in the existing tree after each iteration to ensure the computational complexity and obtain the progressive optimal solution. These two processes are shown in Fig. 1, in which the red node is the new extended node, the white node is the adjacent node in the custom range, that is, the alternative parent node, and the black node is the other nodes in the tree outside the custom range. The advantage of RRT* algorithm is that it reduces the path cost. However, the process of searching for new parent nodes and rewiring repeatedly makes the convergence speed of the algorithm slow, and the path inflection points obtained by the algorithm are too many and not smooth enough.

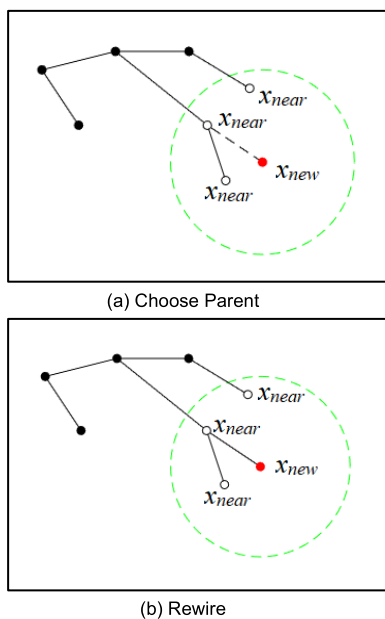


FIGURE 1. Construction of RRT*.

Islam et al. [13] proposed RRT*-Smart, which performs path optimization after finding the initial path and deletes redundant nodes from the found initial path. In addition, it also identifies beacon nodes for path improvement. The second innovation of RRT*-Smart is intelligent sampling, which is biased towards the beacon nodes with optimized paths. Once a shorter path is found, it will perform the path optimization process again to generate new beacon nodes. Therefore, the algorithm accelerates path convergence and reduces path costs. In addition, it uses the bias threshold to select uniform sampling or intelligent sampling. The disadvantage of RRT*-Smart is that the path is not smooth, which leads to an increase in the energy consumption of mobile robots in practical applications.

Gammell et al. [14] proposed Informed-RRT*, which introduced an elliptic sampling strategy to reduce the path cost. Before the initial path is found, the sampling method of the algorithm is the same as that of RRT* algorithm. After the initial path is found, the algorithm sets an ellipse with the starting position and the target position as the focus, and the subsequent sampling will be limited to the ellipse. The smaller the ellipse area, the faster the convergence rate. This algorithm is outstanding in narrow channels, mainly because the probability of random sampling in narrow channels increases. The disadvantage of this algorithm is that it doesn't perform well in complex maze environments. Because of the restriction of the ellipse, the subsequent sampling can't find the path. Similar to this algorithm, Informed RRT*-Connect proposed by Reza Mashayekhi and other scholars seeks the initial path through bidirectional expansion, and the subsequent sampling is limited to an elliptical range [15]. Compared with Informed-RRT*, Informed RRT*-Connect has fewer iterations.

Noreen et al. [16] proposed RRT*-AB, which can quickly locate the target area and improve the computational efficiency and is suitable for large dense sampling space. RRT*-AB proposed an offline planning algorithm based on RRT* to solve the problems of slow convergence and large sampling space. It can quickly locate the target area and improve the calculation efficiency. Through three strategies, connected domain, target-biased bounded sampling, and path optimization, the goal of the algorithm is achieved. Bounded target offset sampling is performed on the boundary of the connected region to find the initial path. Once the path is found, it is gradually optimized by node rejection and centralized bounded sampling. The algorithm adopts new schemes such as connected domain, centralized sampling, and node elimination. By intelligently searching the sampling space, it can quickly converge to the optimal solution, and it also involves a narrow and complex environment. The disadvantage is that there is no smoothing treatment at the inflection point of the path.

Li et al. [17] put forward a method called Fast-RRT*. To reduce the blindness of sampling and improve the efficiency of algorithm planning, a mixed sampling strategy combining target deviation sampling with constrained sampling

was proposed. In constrained sampling, when the constraint conditions are met, the sampling is successful, otherwise, the sampling will continue. Balance the proportion of target sampling and random sampling by appropriate deviation probability. Then, based on the idea of backtracking, the possible parent node set of new nodes is expanded to reduce the path cost. In order to ensure the smoothness of the path, a cubic B-spline curve is used to smooth the path. However, this algorithm does not fully consider the safety of obstacle corner areas.

Jeong et al. [18] put forward RRT*-Quick, which effectively expands the candidate pool of parent nodes by using ancestor nodes to improve the convergence speed. By adding depth parameters and using triangle inequality. Jeong et al. [19] put forward Quick-RRT*, which improves the way of selecting parent nodes. When a new node is added to the tree, the algorithm searches a group of neighboring nodes based on predefined depth parameters to find the parent node of the shortest path to the new node. If there is no collision between paths, the path cost is reduced by triangle inequality. The specific process is shown in Fig. 2. It can be seen intuitively that the process of selecting parent nodes and rewiring in this algorithm is different from RRT*. This algorithm not only straightens the path to the new node but also straightens the path to nearby nodes. This algorithm not only optimizes the selection of parent nodes but also makes the path closer to the edge of obstacles, which is not conducive to the safety of mobile robots and does not consider the smoothness of the path.

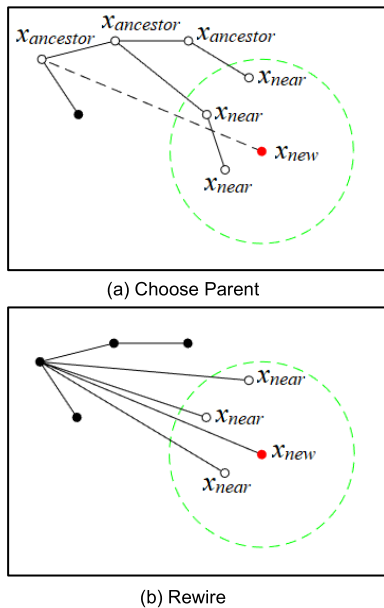


FIGURE 2. Construction of Quick-RRT* (depth = 2).

Orozco-Rosas et al. [20] put forward a hybrid path planning algorithm based on membrane pseudo-bacterial potential field (MemPBPF) is proposed. Membrane-inspired algorithms can reach an evolutionary behavior based on

biochemical processes to find the best parameters for generating a feasible and safe path. This hybridization between membrane computing, the pseudo-bacterial genetic algorithm, and the artificial potential field method provides an outperforming path planning algorithm for autonomous mobile robots. MemPBPF algorithm reaches an evolutionary behavior for finding an efficient path planning in terms of path length and time execution under a parallel computing implementation.

Reference [21] put forward a membrane evolutionary artificial potential field (memEAPF) approach for solving the mobile robot path planning problem is proposed, which combines membrane computing with a genetic algorithm (membrane-inspired evolutionary algorithm with one-level membrane structure) and the artificial potential field method to find the parameters to generate a feasible and safe path. The memEAPF proposal consists of delimited compartments where multisets of parameters evolve according to rules of biochemical inspiration to minimize the path length. The results show that the proposed algorithm is effective and practical. The simulation results show that the smoothness of the path planned by the algorithm needs to be improved.

Orozco-Rosas et al. [22] put forward a method to calculate the feasible path of mobile robots in known and unknown environments by using QAPF learning algorithm. To solve the problem of the slow convergence speed of Q-learning, the concept of partially guided Q-learning is employed wherein, the artificial potential field (APF) method is utilized to improve the classical Q-learning approach. Therefore, the proposed QAPF learning algorithm for path planning can enhance learning speed and improve final performance using the combination of Q-learning and the APF method. Criteria used to measure planning effectiveness include path length, path smoothness, and learning time. The experimental results show that compared with the classical method, QAPF learning algorithm reduces the path cost, improves the path smoothness, and reduces the training time.

In this paper, CAF-RRT* is proposed, which combines the advantages of Quick-RRT* and bidirectional RRT to quickly obtain the initial path, takes it as the optimization object, and optimizes the initial path through the subsequent path optimization strategy and path smoothing strategy.

The main contributions of this paper are as follows,

- (1) When determining the best parent node of a new node, based on the backtracking idea, the earlier parent node is considered to reduce the path cost. And combined with the two-way greedy search strategy, the efficiency of the algorithm is improved.
- (2) A path optimization strategy is proposed to reduce the path cost by using the triangle rule under the premise of ensuring safety.
- (3) A path smoothing strategy that based on Circular Arc Fillet is proposed to realize the path smoothing and improve the path quality.

II. PROPOSED METHOD

This section first defines the variables, symbols, and functions related to the algorithm, then introduces the overall framework of the proposed algorithm, and gives the pseudo-code of the algorithm. Finally, with subsections C and D, it introduces the two main innovations of this paper in detail, the path optimization strategy, and the path smoothing strategy based on the triangle rule.

A. PROBLEM DEFINITION

Let $X = (0, 1)^d$ be the configuration space, X_{obs} be the obstacle region. x_{new} represents the new node obtained by extension. X_{p-obs} is the probability collision region, which closing to the edge of an obstacle and likely to cause a collision. X_{free} be the closure of $X \setminus X_{obs}$. $(X_{free}, x_{init}, X_{goal})$ defines a path planning problem, where $x_{init} \in X_{free}$ is the initial state and $X_{goal} \subset X_{free}$ is the goal area. T_{init} is the set of initial path tree obtained by the algorithm. T_{opt} is the set of optimize path tree obtained by path optimize policy. T_{final} is the set of final path tree obtained by path smoothing policy. A path σ is collision-free if $\forall \tau \in [0, 1], \sigma(\tau) \in X_{free}$. The definition of the optimal path is given by the following formula,

$$\sigma^* = \arg_{\sigma \in \Sigma} \min c(\sigma) \quad (1)$$

$$\text{subject to, } \sigma(0) = x_{init}$$

$$\sigma(1) \in X_{goal}$$

$$\sigma(\tau) \in X_{free}, \quad \forall \tau \in [0, 1] \quad (2)$$

Definition 1: Safety distance is defined as that for any point x on the path $\sigma(\tau), \forall x_{obs} \in X_{obs}, \min \|x - x_{obs}\|_2 > \Delta q$ should be satisfied.

Definition 2: The probability collision region is for $\forall x_{new}$, if $\|x_{new} - X_{obs}\|_2 < \Delta q$, x_{new} is considered to have entered the probability collision region, namely $x_{new} \in X_{p-obs}$.

Make the following statement for the functions in the pseudo-code of this paper. N represents the number of nodes in T_{init} or T_{opt} . V represents a set of points in T_{init} or T_{opt} , function Angle can calculate the angle of two points relative to the coordinate axis according to arctan function. Function Insert means to insert the queue between two points.

B. THE MAIN FRAMEWORK

The overall framework of the algorithm is as follows, firstly, the obstacles are inflated by the inflation strategy to ensure the safety of the planned path. Then, by using the idea of the bidirectional extension of bidirectional RRT and heuristic search, the initial path can be quickly planned, but the generated path has a large number of intermediate nodes. The solution in this paper is combined with the idea of selecting parent nodes by depth parameters mentioned in Quick-RRT*. By detecting the parent node before the parent node, the generation of intermediate nodes can be effectively reduced, the path to the new node and the nearby nodes can be pulled, and the times of collision detection in the subsequent optimization process of the algorithm are also reduced, thus reducing the running time of the algorithm. Then, after the initial path is

obtained, the path optimization strategy is implemented, and the initial path is optimized in two ways, equal proportion and equal distance, by using the triangle rule to reduce the path cost, the specific details of the path optimization strategy are described in subsection C. Finally, through the proposed path smoothing strategy based on circular arc fillet method, the path cost at the middle node of the path can be reduced, a smooth path can be obtained, and the energy consumption of the mobile robot can be reduced. The specific details of the smoothing strategy are described in subsection D. Algorithm 1 is the pseudo-code of the overall flow of the proposed algorithm, among which Ancestry, ChooseParent*, and Rewire* functions are described in detail in [19].

Algorithm 1 CAF-RRT*

```

1  $T_a \leftarrow (V_a, E_a), T_b \leftarrow (V_b, E_b);$ 
2  $\text{map} \leftarrow \text{Inflation}(\text{map});$ 
3 for  $i = 0$  to  $N$  do
4    $x_{rand} \leftarrow \text{Sample}(i);$ 
5    $x_{nearest} \leftarrow \text{Nearest}(T_a, x_{rand});$ 
6    $x_{new} \leftarrow \text{Steer}(x_{nearest}, x_{rand});$ 
7   if  $\text{CollisionFree}(x_{new}, x_{nearest})$  then
8      $X_{near} \leftarrow \text{Near}(x_{new}, T_a);$ 
9      $X_{parent} \leftarrow \text{Ancestry}(T_a, X_{near});$ 
10     $x_{parent} \leftarrow \text{ChooseParent}^*(X_{near} \cup X_{parent}, x_{new});$ 
11     $T_a \leftarrow \text{Rewire}^*(T_a, x_{new}, X_{near});$ 
12     $x_{nearest-b} \leftarrow \text{Nearest}(T_b, x_{new});$ 
13    if  $\text{CollisionFree}(x_{new}, x_{nearest-b})$  &
         $\|x_{new}, x_{nearest-b}\|_2 < \sigma$  then
14      return  $T_{init};$ 
15    else
16       $\text{SWAP}(T_a, T_b);$ 
17    end if
18  end if
19 end for
20  $T_{opt} \leftarrow \text{Path optimization}(T_{init}, \Delta e, \Delta q, p);$ 
21  $T_{final} \leftarrow \text{Path smoothing}(T_{opt}, w);$ 
22 return  $T_{final};$ 

```

C. PATH OPTIMIZATION POLICY

After the initial path is obtained by combining Quick-RRT* with bidirectional RRT, the path optimization strategy of equal distance and equal proportion is carried out alternately for the initial path, which can reduce the path cost without affecting the path safety. The main steps are to select all the intermediate nodes and two adjacent edges of the path in turn and construct triangles for optimization. For example, for any intermediate node V_i , take the front and back nodes to build triangles named as $\Delta V_{i-1} V_i V_{i+1}$. In the equal-distance method, define a length Δe , from V_i to V_{i-1} , take point V'_{i-1} at interval Δe , then, from V_i to V_{i+1} , take point V'_{i+1} at interval Δe , to build a new triangle named as $\Delta V'_{i-1} V_i V'_{i+1}$. In the equal-proportion method, define the ratio $p(p \in (0, 1))$, from V_i to V_{i-1} , take point V''_{i-1} at interval $\|V_i, V_{i-1}\|_2 * p$, from V_i to V_{i+1} , take point V''_{i+1} at interval $\|V_i, V_{i+1}\|_2 * p$,

to build a new triangle named as $\Delta V''_{i-1}V_iV''_{i+1}$. Collision detection for edge $V'_{i-1}V'_{i+1}$ or $V''_{i-1}V''_{i+1}$, take points on the edge $V'_{i-1}V'_{i+1}$ or $V''_{i-1}V''_{i+1}$ to detect the node security policy. According to the triangle rule, multiple cycles are carried out to reduce the path cost. Because points are taken on the straight line segment many times, redundant nodes may be generated in the sub-path after this process, which needs to be eliminated, otherwise, it will affect the smoothing process of the fillet method. The specific steps in this section are shown in the pseudo-code of Algorithm 2, and the comparison of the effects is shown in Fig. 3.

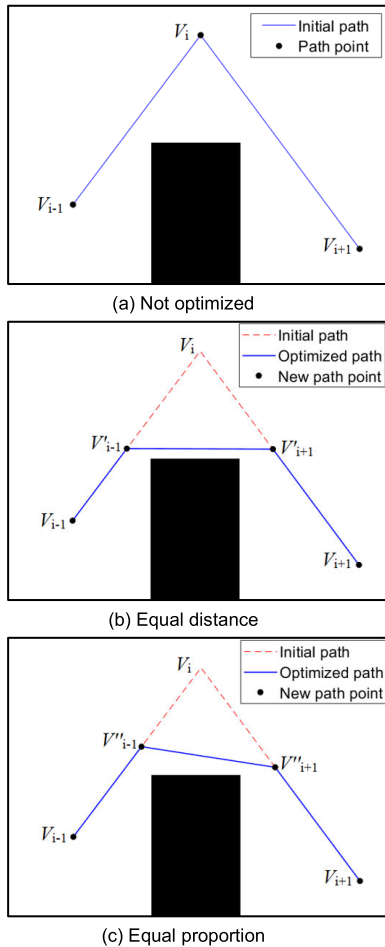


FIGURE 3. Optimized path.

D. PATH SMOOTHING POLICY

After the path optimization strategy, the path cost of the obtained path is effectively reduced, but the obtained path is not smooth enough, especially at the corner, so it is necessary to smooth the path to further improve the path quality. In this paper, a path smoothing strategy based on the circular arc fillet method is proposed and applied to global path planning [23]. The circular arc fillet method has good geometric properties and is widely used in engineering and architectural engineering. Its remarkable advantage is that it can be applied to straight lines with any intersection angle. Therefore, the

Algorithm 2 Path Optimization ()

Input: $T_{init}, \Delta e, \Delta q, p$

Output: T_{opt}

```

1 for j = 1 to 2 do
2   for i = 2 to  $N_{init} - 1$  do
3      $l_1 = \|V_i, V_{i-1}\|_2; l_2 = \|V_i, V_{i+1}\|_2;$ 
4      $\alpha_1 = \text{Angle}(V_i, V_{i-1}); \alpha_2 = \text{Angle}(V_i, V_{i+1});$ 
5      $V'_{(i-1)x} = V_{ix} + \Delta e \times \cos(\alpha_1); V'_{(i-1)y} = V_{iy} +$ 
       $\Delta e \times \sin(\alpha_1);$ 
6      $V'_{(i+1)x} = V_{ix} + \Delta e \times \cos(\alpha_2); V'_{(i+1)y} = V_{iy} +$ 
       $\Delta e \times \sin(\alpha_2);$ 
7     if CollisionFree ( $V'_{i-1}, V'_{i+1}$ ) then
8       delete  $V_i;$ 
9        $(V_{i-1}, V_{i+1}) \leftarrow \text{Insert}(V'_{i-1}, V'_{i+1});$ 
10    end if
11  end for
12 return  $T_{opt};$ 
13 for i = 2 to  $N_{opt} - 1$  do
14    $l_1 = \|V_i, V_{i-1}\|_2; l_2 = \|V_i, V_{i+1}\|_2;$ 
15    $\alpha_1 = \text{Angle}(V_i, V_{i-1}); \alpha_2 = \text{Angle}(V_i, V_{i+1});$ 
16    $V''_{(i-1)x} = V_{ix} + l_1 \times p \times \cos(\alpha_1); V''_{(i-1)y} = V_{iy} +$ 
      $l_1 \times p \times \sin(\alpha_1);$ 
17    $V''_{(i+1)x} = V_{ix} + l_2 \times p \times \cos(\alpha_2); V''_{(i+1)y} = V_{iy} +$ 
      $l_2 \times p \times \sin(\alpha_2);$ 
18   if CollisionFree ( $V''_{i-1}, V''_{i+1}$ ) then
19     delete  $V_i;$ 
20      $(V_{i-1}, V_{i+1}) \leftarrow \text{Insert}(V''_{i-1}, V''_{i+1});$ 
21   end if
22 end for
23 return  $T_{opt};$ 
24 end for
25 for k = 2 to  $N_{opt} - 1$  do
26   if CollisionFree ( $V_{k-1}, V_{k+1}$ ) then
27     delete  $V_k;$ 
28     Update  $T_{opt};$ 
29   end if
30 end for
31 return  $T_{opt};$ 

```

fillet method is a good curve fitting tool, which smoothed line segments by converting them into arcs. The specific process is as follows.

1) FIND THE BEST SEGMENTATION POINT

After the given stator path smoothed its corners, there should not be any sharp edges left. This depends not only on the radius of the circle but also on the selection of the cutting point. In Fig. 4(a), there are two circles placed on the given edge of the sub-path. Although they have the same radius, the cutting point will affect the smoothness of the new path formed. After cutting sub-paths according to circle C and circle D respectively, a new sub-path is formed, as shown in Fig. 4(b) and Fig. 4(c). In Fig. 4(c), points E and F are not smooth enough, while in Fig. 4(b), the arc can smoothly

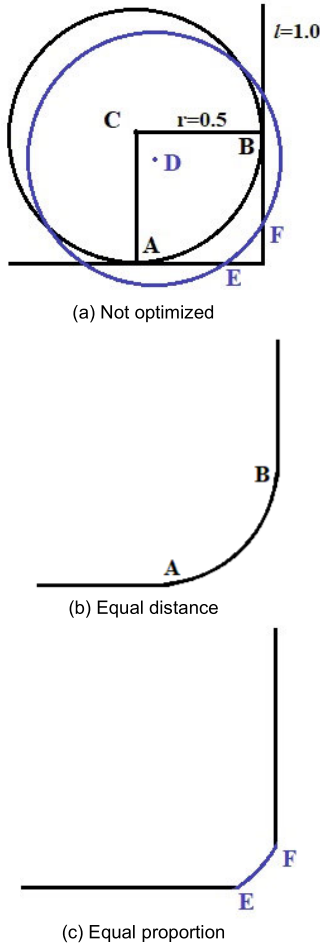


FIGURE 4. Circle C and circle D cut the edges formed by sub-paths from different points.

connect the two sides, A and B are the best cutting points when the length of the line segment is 1. When r remains unchanged, modifying the length of the line segment will change the positions of points A and B, the specific algorithm for finding the best cutting point is shown in the pseudo-code of algorithm 3. Therefore, the arc of the optimal circle is consistent with the smoothness of the edge after cutting, and it must touch the edge with a tangent, but not intersect the edge, to avoid additional corners.

2) CALCULATE THE OPTIMAL RADIUS

In Fig. 5, triangles CBA and CBA' are congruent, CA and CA' are the radius of a circle, and the angles of CBA and CBA' are the same because triangles are congruent. Therefore, $2\angle\beta = \angle\alpha$. The length of AB and A'B is half of the shorter side (can be modified to other proportions). To find the radius of a smooth circle, you need to find the angle between the edges,

$$\alpha_i = \text{Arctan} \frac{y_i - y_{i+1}}{x_i - x_{i+1}} - \text{Arctan} \frac{y_{i+2} - y_{i+1}}{x_{i+2} - x_{i+1}} \quad (3)$$

If the angle is negative, then,

$$\alpha_{i_{new}} = 2\pi\alpha_i \quad (4)$$

Algorithm 3 Path Smoothing ()

Input: $T_{opt,w}$

Output: T_{final}

```

1 for  $i = 2$  to  $N_{opt} - 1$  do
2    $l_1 = \|V_i, V_{i-1}\|_2, l_2 = \|V_i, V_{i+1}\|_2$ ;
3    $\beta_1 = \text{Angle}(V_i, V_{i-1}), \beta_2 = \text{Angle}(V_i, V_{i+1})$ ;
4    $l = \min(l_1, l_2)/w$ ;
5    $A_x = V_{ix} + l \times \cos(\beta_1), A_y = V_{iy} + l \times \sin(\beta_1)$ ;
6    $A'_x = V_{ix} + l \times \cos(\beta_2), A'_y = V_{iy} + l \times \sin(\beta_2)$ 
7    $r = |l \times \tan((\beta_1 - \beta_2)/2)|$ 
8    $C_x = A_x + r \times \cos(\beta_1 + \pi/2); C_y = A_y + r \times \sin(\beta_1 + \pi/2)$ ;
9    $C'_x = A_x + r \times \cos(\beta_1 - \pi/2); C'_y = A_y + r \times \sin(\beta_1 - \pi/2)$ ;
10   $C''_x = A'_x + r \times \cos(\beta_2 + \pi/2); C''_y = A'_y + r \times \sin(\beta_2 + \pi/2)$ ;
11   $C'''_x = A'_x + r \times \cos(\beta_2 - \pi/2); C'''_y = A'_y + r \times \sin(\beta_2 - \pi/2)$ ;
12  if  $(C'_x = C''_x \ \&\& \ C'_y = C''_y) \parallel (C'_x = C'''_x \ \&\& \ C'_y = C'''_y)$  then
13     $C = C'$ ;
14  else
15     $C = C$ ;
16  end if
17  delete  $V_i$ ;
18   $(V_{i-1}, V_{i+1}) \leftarrow \text{Insert}(A, AA', A')$ ;
19 end for
20 return  $T_{final}$ ;

```

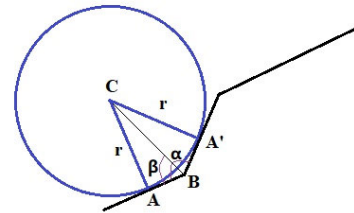


FIGURE 5. Find the radius of the circle.

To calculate the radius of a circle, you need to calculate the angle between the vector from the center of the circle to the node and the edge. At the i th node, this angle is,

$$\beta_i = \alpha_i/2 \quad (5)$$

With the help of trigonometric function, the radius of the smooth circle between edges can be found as,

$$r_i = \tan(\beta_i) * |AB| \quad (6)$$

Then, the center coordinates can be calculated by cutting points A and A', radius r and angle β . There are four center coordinates in total, which are located on both sides of A and A' respectively, as shown in Fig. 6, where the center coordinates of circle C and circle C'' are the same, which is the desired cutting circle. The specific algorithm is shown in the pseudo-code of algorithm 3.

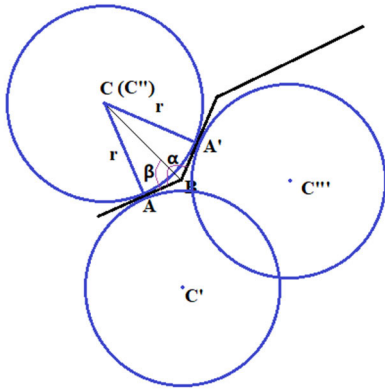


FIGURE 6. Calculate center coordinates.

3) PATH COST FUNCTION EQUATION

This section discusses the change of path cost before and after the path smoothing strategy. The path cost measure before the path smoothing strategy is Euclidean distance. Let any two points $V_i(x_i, y_i)$ and $V_{i+1}(x_{i+1}, y_{i+1})$ in the set of path points V be set, then the path cost between V_i and V_{i+1} can be calculated by (7), while the initial path cost and the path cost after the path optimization strategy are accumulated by the Euclidean distance in the set.

$$\|V_i, V_{i+1}\|_2 = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \quad (7)$$

Path smoothing strategy is to optimize every point in set T_{opt} except the starting node and target node. Therefore, we only discuss one optimization and calculate this process many times, and then we can get the reduction of path cost through path smoothing strategy. In Fig. 5, let the node to be optimized be B, the path lengths of the two terminals of the node are l_{B1} and l_{B2} , respectively, and the length l_{B1} is less than l_{B2} . We select half of the l_{B1} , that is, the midpoint of the shorter sub-path, as the best cutting point, so as to ensure that the front and rear nodes can find suitable cutting points. The radius of the cutting circle can be calculated by (6) as follows:

$$r_B = \frac{1}{2} \tan(\beta) * l_{B1} \quad (8)$$

The length of the smooth arc is:

$$AA' = \frac{(\pi - \alpha)\pi r_B}{180} \quad (9)$$

It can be calculated that the path cost reduction after single smoothing is:

$$\Delta = l_{B1} - \frac{(\pi - \alpha)\pi r_B}{180} \quad (10)$$

By analogy, the total reduction of the path cost after the path smoothing strategy can be calculated.

III. SIMULATION RESULTS

In this section, MATLAB simulation is carried out to prove the effectiveness of the proposed algorithm. In the

first experiment, the overall performance of the proposed algorithm under different iterations of the same map is verified. In the second experiment, RRT*, Quick-RRT*, Fast-RRT*, Informed RRT*-Connect and CAF-RRT* algorithms are compared in maze and clutter maps to highlight their advantages. Finally, CAF-RRT* is tested several times in a complex maze map to verify the adaptability of the proposed algorithm [24]. The simulation was carried out on a computer with Intel (R) Core (TM) i5-10600 KCPU @ 4.10 GHz and 32GB memory. The operating system was 64-bit Windows S10, and the MATLAB version was R2021b.

A. RELATED PARAMETER SETTINGS

This section will discuss the settings of related parameters of the proposed algorithm, and analyze the impact of changing them on the path planning results. Relevant parameters include depth, distance, and proportion of path optimization strategy, cutting point position of path smoothing strategy, iteration times, and expansion radius.

The depth is discussed in detail in [19]. Although the larger depth improves the quality of the solution, it also increases the number of calls to the Collisionfree procedure which strongly affects the running time. Therefore, the depth parameter in this paper is set to 2 as in [19].

Changing the distance and proportion of the path optimization strategy have similar effects. The bigger the setting, the greedier the algorithm will be, and the fewer times Collisionfree procedure will be called, but too high will affect the path optimization strategy's effect of reducing the path cost. The smaller the setting, the more times the Collisionfree procedure is called, but the better the effect of reducing the path cost. We tested the effects of these two parameters on the running time and path cost of the algorithm on several maps, the test results of one of the maps are shown in Table 1. After comprehensive consideration, set the distance parameter to 10 units and the scale parameter to 3%.

TABLE 1. Comparison of different distance and proportion parameters.

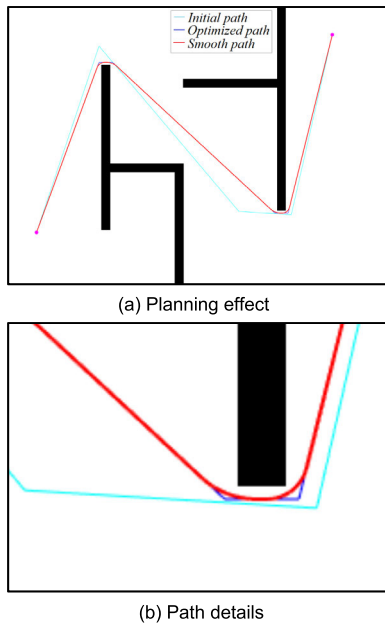
distance	proportion	Running time(s)	Path cost
10	1%	0.68	1850.46
10	3%	0.47	1851.07
20	1%	0.46	1852.51
20	3%	0.44	1855.25

The cutting point position of the path smoothing strategy is set to 1/2 of the short side, that is, the midpoint position. If this value is too large, it will fail to smooth the path continuously by the circular arc fillet method, while if it is too small, it will lead to no obvious smoothing effect.

The maximum number of iterations is 3,500, and the expansion radius of obstacles is 5 units.

TABLE 2. Comparison of different iteration times.

Iterations	Number of samples	Run time(s)	Initial path cost	Optimize path cost	Smooth path cost
1500	626.97	0.64	2025.68	1854.21	1840.85
2000	620.63	0.62	2007.20	1854.45	1841.13
2500	625.40	0.65	2024.67	1854.50	1841.31
3000	619.45	0.63	2016.39	1854.49	1841.04

**FIGURE 7.** Path display in different stages.

B. THE OVERALL PERFORMANCE OF PROPOSED METHOD

The proposed algorithm is run 100 times with different iterations in the maze map, and the iterations of 1,500, 2,000, 2,500, and 3,000 are selected to study the influence of different iterations on the running parameters. The effect of certain planning is shown in Fig. 7(a), and Fig. 7(b) shows the path details. It can be seen that the final path (red path) is the best path. The comparison results are shown in Table 2. It can be seen that although the number of iterations is different, the number of iterations required to get the initial path is less than 630, which indicates that the proposed algorithm needs fewer iterations to find the initial path. Under different iterations, although the initial path cost obtained by the algorithm is high, the path cost is reduced by 8.47%, 7.61%, 8.41%, and 8.03% after the optimization strategy is implemented. And after implementing the path smoothing strategy, the path cost is reduced again. The cost of the initial path and the final path is reduced by 9.12%, 8.27%, 9.06%, and 8.70% respectively. We have also conducted experiments on other maps, and the experimental results are consistent [24].

C. COMPARED WITH RELATED WORKS

In this section, two representative map environments are selected, RRT*, Quick-RRT*, Fast-RRT*, Informed RRT*-Connect and CAF-RRT* algorithms are respectively run

100 times in two maps, which proves the advantages of the proposed algorithm in running time and route cost, and the quality of the planned route is high.

Fig. 8 and Fig. 9 show the simulation results of RRT*, Quick-RRT*, Fast-RRT*, Informed RRT*-Connect, and CAF-RRT* algorithms in two environments. It can be seen intuitively from the figure that, compared with RRT*, Quick-RRT*, and Informed RRT*-Connect algorithms, the path planned by the proposed algorithm is far away from obstacles, thus ensuring the safety of mobile robots, smoothing the inflection point of the path and improving the quality of the path. Although Fast-RRT* also considers the safety and smoothness of the path, its mixed sampling strategy will cause random points to move to the target point, resulting in some map planning failures under the same number of iterations, and may even fall into local minima. Because of the consideration of path safety, the path cost of Fast-RRT* in the maze map is high. In addition, in maze map, because the optimal path length between the start position and the target area is larger than the map, the ellipse sampling area drawn by Informed RRT*-Connect is larger, and the sampling strategy of Informed RRT*-Connect is invalid. Therefore, in maze map, Informed RRT*-Connect is equivalent to RRT*-Connect algorithm, and the path cost is similar to RRT*.

Table 3 is the numerical comparison of the path cost of the above algorithms in 100 simulations in both maps. In maze map, the path cost of the proposed algorithm is reduced by 2.18% compared with RRT*, 1.28% compared with Quick-RRT*, 1.72% compared with Fast-RRT* and 1.97% compared with Informed RRT*-Connect. In clutter map, the path cost of the proposed algorithm is reduced by 0.33%, 0.01%, 1.38%, and 0.11% respectively compared with RRT*, Quick-RRT*, Fast-RRT*, and Informed RRT*-Connect. In clutter map, the path cost of CAF-RRT* is similar to that of other algorithms because other algorithms do not consider the safety of robots. When the algorithm is applied in practice,

TABLE 3. Length of planned path.

algorithm	maze	clutter
RRT*	1882.08	1215.74
Quick-RRT*	1862.11	1211.76
Fast-RRT*	1873.29	1228.67
Informed RRT*-Connect	1878.06	1213.04
CAF-RRT*	1841.10	1211.70

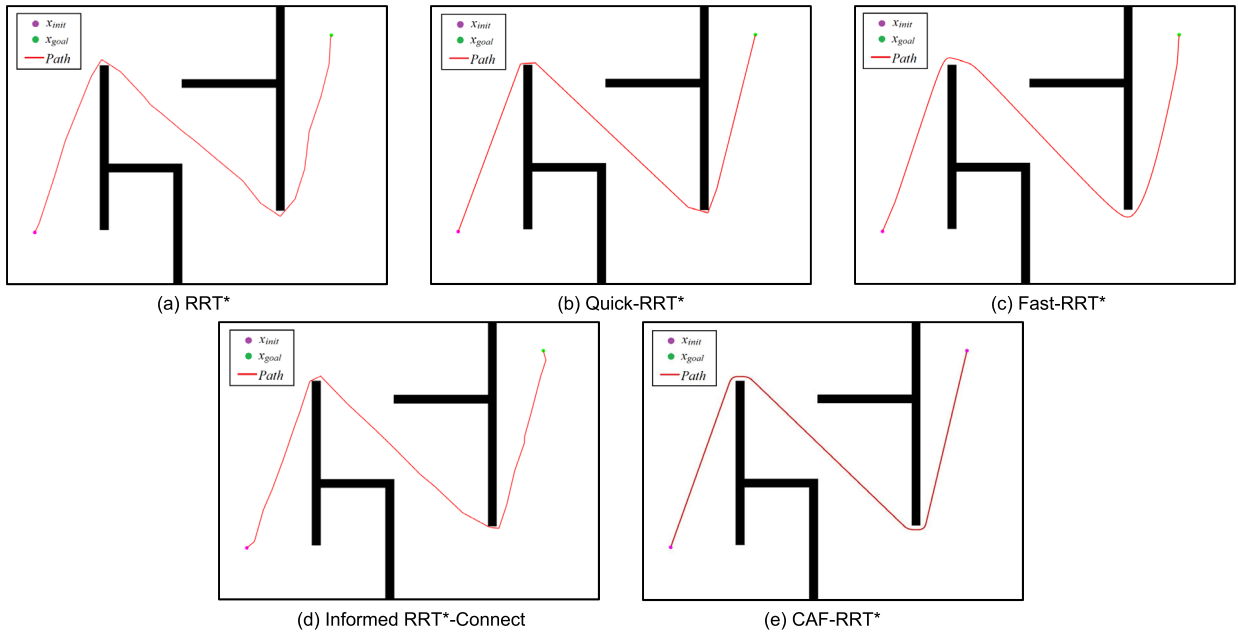


FIGURE 8. Simulation results (maze).

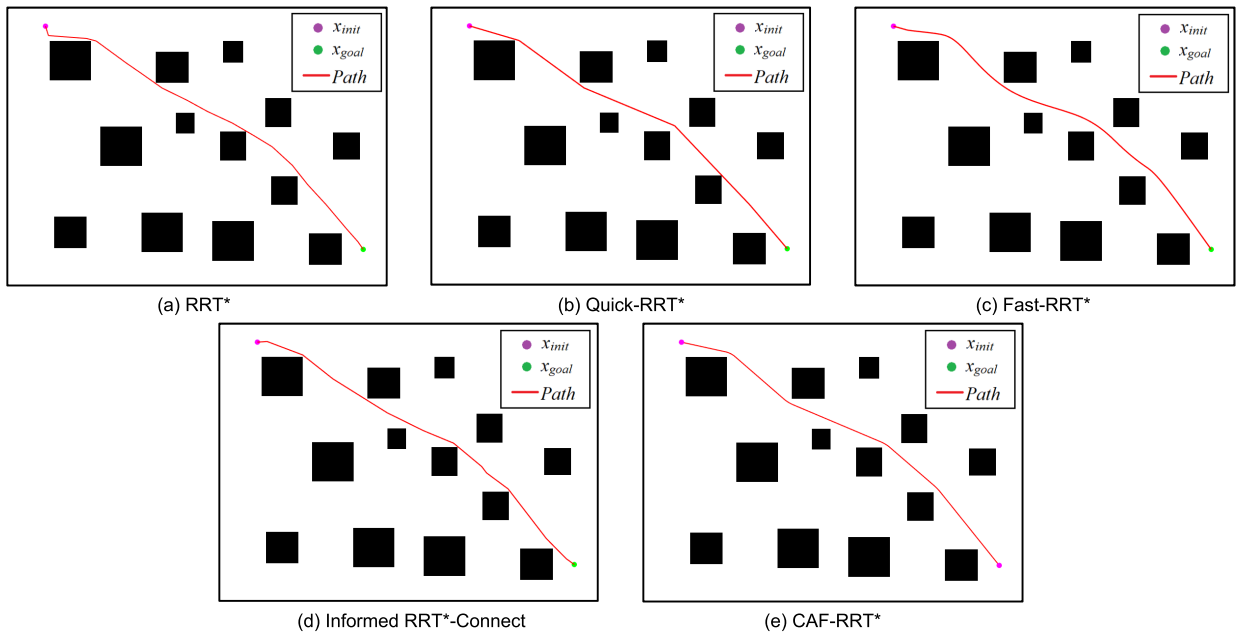


FIGURE 9. Simulation results (clutter).

it is necessary to add an inflation layer, and the path cost of the proposed algorithm will be lower than that of other algorithms. Therefore, CAF-RRT* can reduce the path cost, and the path planned by CAF-RRT* is safer and smoother, thus improving the path quality.

The box diagram of the path cost of the above algorithms in two environments is shown in Fig. 10. It can be seen that the difference between the maximum and minimum path cost of CAF-RRT* in the two graphs is smaller than that of other algorithms, which indicates that the path cost fluctuation

of this algorithm for multiple planning is smaller, and the difference between the upper and lower quartiles is smaller than that of other algorithms, which indicates that the path cost of this algorithm is more stable.

To compare the running time, RRT*, Quick-RRT*, Fast-RRT*, Informed RRT*-Connect, and CAF-RRT* algorithm is run 100 times on two maps respectively. In maze map, the comparison of the running time values of the above algorithms is shown in Table 4. The average convergence time of CAF-RRT* is reduced by 97.25% compared with RRT*,

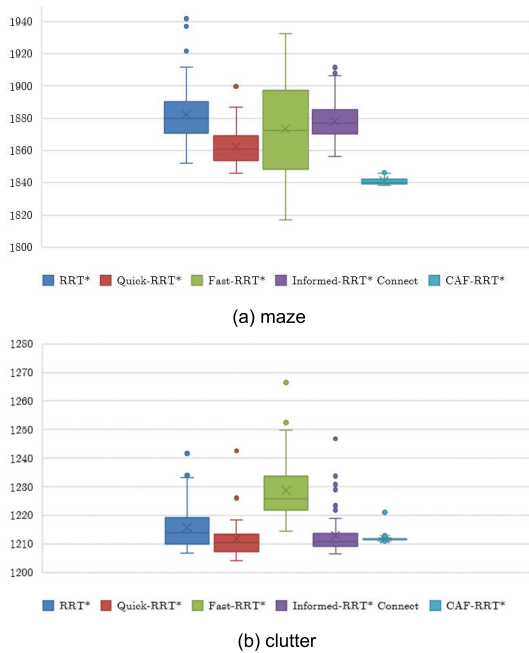


FIGURE 10. Stability comparison.

TABLE 4. Stability of running time in maze.

algorithm	Mean(s)	Standard deviation	Max(s)	Min(s)
RRT*	23.31	2.06	28.62	18.41
Quick-RRT*	17.18	1.94	23.01	12.98
Fast-RRT*	16.49	6.12	38.20	4.68
Informed RRT*-Connect	40.62	5.51	50.84	26.64
CAF-RRT*	0.64	0.19	1.66	0.36

96.25% compared with Quick-RRT*, 96.12% compared with Fast-RRT* and 98.42% compared with Informed RRT*-Connect. Therefore, CAF-RRT* can effectively reduce the running time of RRT* correlation algorithm, and the standard deviation of the running time of CAF-RRT* is the smallest among all algorithms, which indicates that the algorithm is more stable and the data distribution is more concentrated in multiple runs.

Fig. 11 is a box diagram of the running time of CAF-RRT* in the maze map. It can be seen from the diagram that the average running time of CAF-RRT* is smaller than that of other algorithms, and the difference between the upper and lower quartiles is small, so the running time of CAF-RRT* is more stable.

Table 5 shows the numerical comparison of the running time of the above algorithms in clutter map. As can be seen from the table, the average running time of CAF-RRT* is reduced by 99.32% compared with RRT*, 98.38% compared with Quick-RRT*, 99.54% compared with Fast-RRT* and 97.67% compared with Informed RRT*-Connect. In this map,

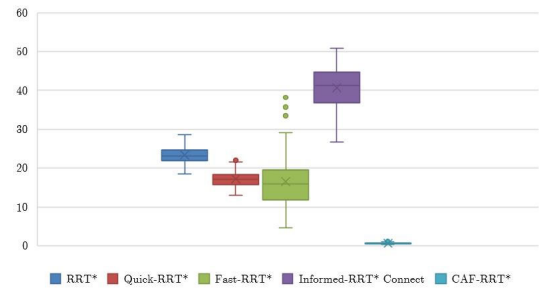


FIGURE 11. Running time comparison(maze).

TABLE 5. Stability of running time in clutter.

algorithm	Mean(s)	Standard deviation	Max(s)	Min(s)
RRT*	23.65	0.56	25.29	22.59
Quick-RRT*	9.87	0.68	10.90	7.96
Fast-RRT*	34.71	8.58	61.38	20.23
Informed RRT*-Connect	6.86	1.47	11.30	4.65
CAF-RRT*	0.16	0.02	0.20	0.10

CAF-RRT* still has obvious advantages in running time, and the standard deviation is smaller.

The running time box diagram of the above algorithm in the clutter map is shown in Fig. 12. It can be seen that the difference between the average and the upper and lower quartiles of CAF-RRT* is smaller than other algorithms, and the number of outliers is small. so, the running time of CAF-RRT* in this map is shorter and the algorithm is more stable.

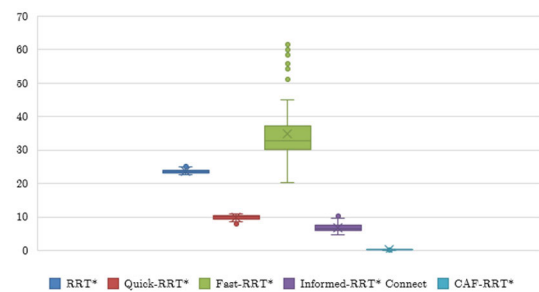


FIGURE 12. Running time comparison(clutter).

D. ADAPTABILITY EXPERIMENT

Adaptability is an important criterion to measure the path planning algorithm. In this section, we simulate RRT*, Quick-RRT*, Fast-RRT*, Informed RRT*-Connect, and CAF-RRT* algorithms in a maze-tough map to compare their adaptability. Compared with maze map and clutter map, maze-tough map has more obstacles, higher map complexity, and better adaptability of the algorithm. In this experiment, the number of iterations is set to 50,000, and other parameters remain unchanged. The listed algorithms are simulated

TABLE 6. Adaptive experimental results.

algorithm	Average running time(s)	Average path cost	Number of nodes
RRT*	539.40	2528.28	58
Quick-RRT*	497.31	2475.50	50
CAF-RRT*	57.65	2472.85	47

several times. Table 6 shows the statistics of simulation results, and Figure 13 shows the simulation results. Because the planning success rate of Informed RRT*-Connect and Fast-RRT* are low under the current iteration number, no comparison is made. The red line or green line is the extension result of two RRT trees, and the blue line is the final path. From the simulation results, it can be seen that the running time of the proposed algorithm is shorter, the planned final path can keep a certain safe distance from obstacles, and the inflection point of the path is relatively smooth. Therefore, compared with similar algorithms, CAF-RRT* has good adaptability.

IV. EXPERIMENT VERIFICATION BASED ON ROS PLATFORM

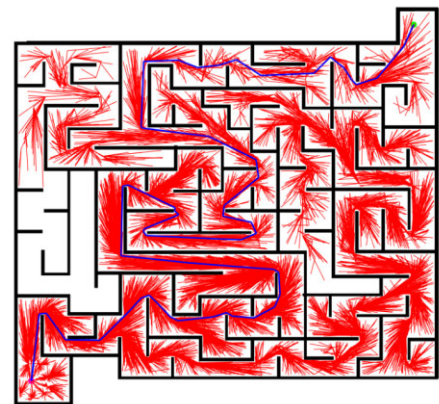
The virtual machine is configured as Ubuntu16.04LTS, the processor is AMD Ryzen 7 5800H, and the running memory is 8G. The experiment was carried out under the open-source robot operating platform ROS, and SLAM technology was used in Rviz to locate the robot independently and build the map. The third-generation TurtleBot3 dual-wheel differential drive robot was selected as the experimental object. Fig. 14(a) is the actual TurtleBot3 robot, the main functional specifications of the TurtleBot3 are the maximum translational velocity: 0.22 m/s, and the maximum rotational velocity: 162.7 deg/s. Fig. 14(b) is the TurtleBot3 robot model established in Gazebo for the ROS simulation experiment. Because the effectiveness of the path security strategy in the algorithm proposed in this paper is not clear, in order to avoid potential security risks, we should first conduct a mapping test in the virtual environment, and then conduct experiments in the real environment.

A. SIMULATION IN GAZEBO

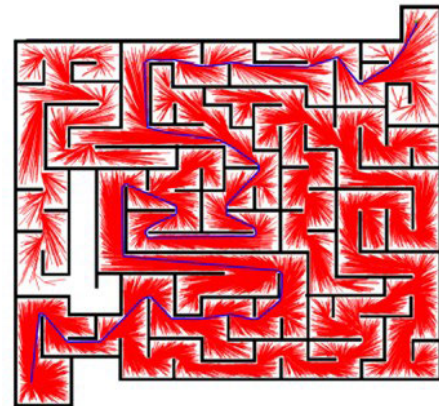
Set up an environment map in Gazebo, and the simulation environment is set up with reference to the frame map. Fig. 15 shows the navigation process of Turtlebot3 in the Gazebo. It can be seen from Fig. 15(b) that the path planned by the proposed algorithm can keep a safe distance from obstacles, thus ensuring the safety of the path and making the path smoother. The path planned in Gazebo is similar to the simulation result in Matlab.

B. SIMULATION IN REAL ENVIRONMENT

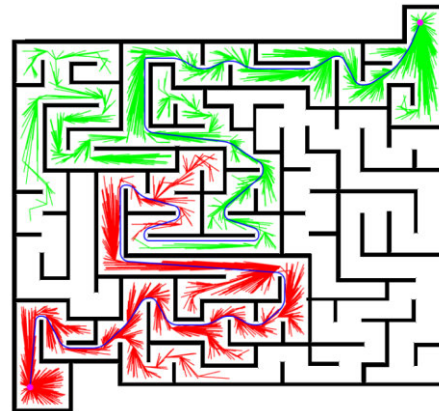
In this section, several real scenes are constructed, and the performance of the algorithm is verified by comparative



(a) RRT*



(b) Quick-RRT*



(c) CAF-RRT*

FIGURE 13. Adaptability experiment of algorithm in maze-tough.

experiments. RRT*, Quick-RRT*, Fast-RRT*, Informed RRT*-Connect, and CAF-RRT* algorithms are used as global path planning algorithms respectively, with the inflation radius of 0.2m, the Turtlebot3 line speed of 0.1m/s and the angular velocity of 40deg/s.

The starting position of the first map is shown in Fig. 16(a), the target position is shown in Fig. 16(d), and the obstacles are constructed as a small maze. Turtlebot3 moves around the map by computer control, and different planning algorithms are deployed as global path planning plug-ins by

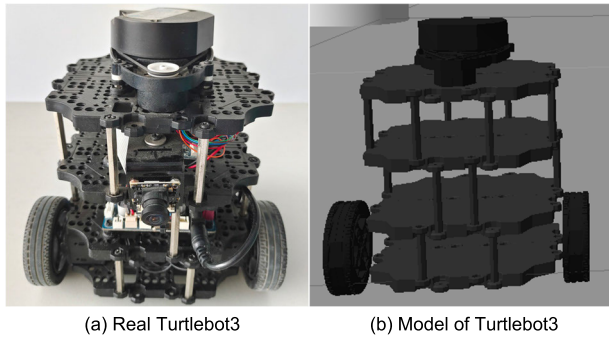


FIGURE 14. Turtlebot3 mobile robot.

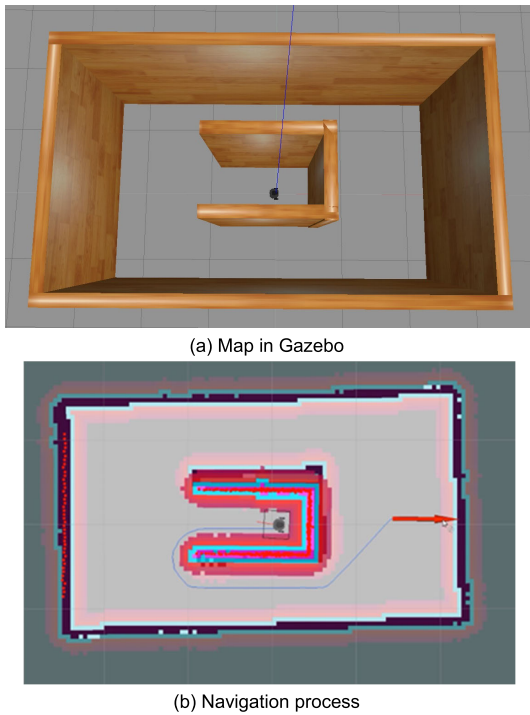


FIGURE 15. Virtual environment navigation process.

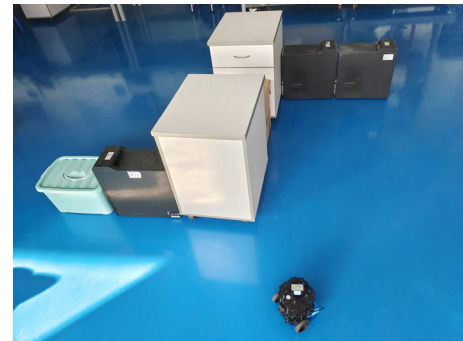
TABLE 7. Simulation results of map 1.

algorithm	Path planning time(s)	Simulation time(s)	Path length (m)
RRT*	11.79	70.38	5.72
Quick-RRT*	7.98	64.67	5.57
Fast-RRT*	7.71	63.79	5.62
Informed RRT*-Connect	17.23	74.53	5.67
CAF-RRT*	0.32	54.21	5.33

SLAM positioning and map building. The simulation results of each algorithm on this map are shown in Table 7, and the simulation process of the proposed algorithm is shown in Fig. 16.



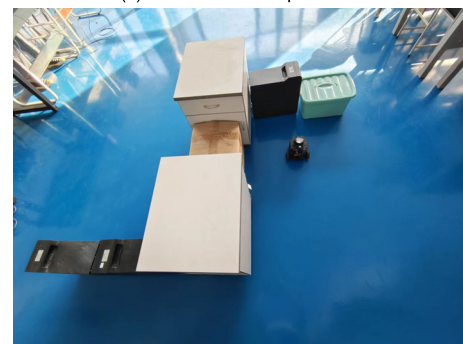
(a) CAF-RRT* in map 1



(b) CAF-RRT* in map 1



(c) CAF-RRT* in map 1



(d) CAF-RRT* in map 1

FIGURE 16. Simulation process of map 1.

The starting position of the second map is shown in Fig. 17(a), the target position is shown in Fig. 17(b), and the obstacle is set as a cluttered warehouse. Once again, Turtlebot3 is mapped and navigated, and different planning algorithms are deployed as global path planning plug-ins. The simulation results of each algorithm on the map are shown in



(a) CAF-RRT* in map 2



(b) CAF-RRT* in map 2



(c) CAF-RRT* in map 2



(d) CAF-RRT* in map 2

FIGURE 17. Simulation process of map 2.

Table 8, and the simulation process of the proposed algorithm is shown in Fig. 17.

Unlike the Matlab simulation environment, it can be seen from the above simulation results of the real environment that the CAF-RRT* algorithm has shorter path planning time and less cost for the planned paths in different maps compared

TABLE 8. Simulation results of map 2.

algorithm	Path planning time(s)	Simulation time(s)	Path length (m)
RRT*	9.39	62.39	5.32
Quick-RRT*	4.11	56.33	5.17
Fast-RRT*	11.27	63.53	5.21
Informed RRT*-Connect	3.91	60.26	5.19
CAF-RRT*	0.16	49.61	4.72

to other RRT* algorithms under the premise of ensuring the safety of the mobile robot, which makes the paths planned by Turtlebot3 in the real experimental environment more satisfactory.

V. CONCLUSION

In order to obtain the planning path with progressive optimal length, this paper proposes multiple optimization strategies based on Quick-RRT* algorithm and bidirectional RRT algorithm, aiming at the problem that the traditional RRT algorithm does not consider the path safety and smoothness. The proposed path smoothing strategy based on the circular arc fillet method can effectively reduce the energy consumption of mobile robots and enhance the practicability of the algorithm.

We employed different test scenarios to evaluate CAF-RRT* algorithm, and the results show that the proposed algorithm achieves the three requirements of path planning: path cost, security, and smoothness, so the proposed algorithm has strong adaptability and practicality. The limitation of the proposed algorithm is the need to test the two parameters, distance and scale, of the path optimization strategy to determine the better choice of parameters. Therefore, a future improvement would be to combine these two parameters with information from the planning map and perform adaptive calculations to determine the parameter values.

In addition, there are several possible directions to improve this work in the future. Firstly, only one mobile robot is considered in this paper, so it may be interesting to pay attention to the cooperative path planning of multi-robots in a dynamic uncertain environment. Secondly, by combining reinforcement learning and neural network, it may be possible to solve a wider range of path planning problems. Lastly, the algorithm in this paper can be extended to work in 3D space, which is useful for many applications such as information collection (i.e., drones), disaster relief, and exploration.

REFERENCES

- [1] H. Zhaozheng, X. Cong, Z. Zhe, and D. Zewu, "Robot localization from planned path constraints," *J. Electron. Inf. Technol.*, vol. 44, pp. 1–10, Apr. 2022.
- [2] X. Ruan, J. Zhou, J. Zhang, and X. Zhu, "Robot goal guide RRT path planning based on sub-target search," *Control Decis.*, vol. 35, pp. 2543–2548, Jun. 2020.

- [3] J. Qi, H. Yang, and H. Sun, "MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment," *IEEE Trans. Ind. Electron.*, vol. 68, no. 8, pp. 7244–7251, Aug. 2021.
- [4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
- [5] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998, pp. 98–111.
- [6] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. Robot. Autom.*, vol. 14, no. 1, pp. 166–171, Feb. 1998.
- [7] I. Noreen, A. Khan, and Z. Habib, "A comparison of RRT, RRT* and RRT*-smart path planning algorithms," *Int. J. Comput. Sci. Netw. Secur.*, vol. 16, no. 10, p. 20, 2016.
- [8] J. J. Kuffner, Jr., and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Apr. 2000, pp. 995–1001.
- [9] Z. Wu, Z. Meng, W. Zhao, and Z. Wu, "Fast-RRT: A RRT-based optimal path finding method," *Appl. Sci.*, vol. 11, no. 24, p. 11777, Dec. 2021.
- [10] J.-G. Kang and J.-W. Jung, "Post triangular rewiring method for shorter RRT robot path planning," 2021, *arXiv:2107.05344*.
- [11] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, and I. Ahmedy, "Hybrid RRT: A semi-dual-tree RRT-based motion planner," *IEEE Access*, vol. 8, pp. 18658–18668, 2020.
- [12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
- [13] F. Islam, "RRT*-smart: Rapid convergence implementation of RRT* towards optimal solution," in *Proc. IEEE Int. Conf. Mechatron. Automat.*, Jun. 2012, pp. 1651–1656.
- [14] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.
- [15] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy, and I. Ali, "Informed RRT*-connect: An asymptotically optimal single-query path planning method," *IEEE Access*, vol. 8, pp. 19842–19852, 2020.
- [16] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, "Optimal path planning in cluttered environment using RRT*-AB," *Intell. Service Robot.*, vol. 11, no. 1, pp. 41–52, Jan. 2018.
- [17] Q. Li, J. Wang, H. Li, B. Wang, and C. Feng, "Fast-RRT*: An improved motion planner for mobile robot in two-dimensional space," *IEEE Trans. Electr. Electron. Eng.*, vol. 17, no. 2, pp. 200–208, Feb. 2022.
- [18] I. B. Jeong, S. J. Lee, and J. H. Kim, "RRT*-quick: A motion planning algorithm with faster convergence rate," in *Robot Intelligence Technology and Applications 3*. Cham, Switzerland: Springer, 2015, pp. 67–76.
- [19] I.-B. Jeong, S.-J. Lee, and J.-H. Kim, "Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Syst. Appl.*, vol. 123, pp. 82–90, Jun. 2019.
- [20] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019.
- [21] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput. J.*, vol. 77, pp. 236–251, Apr. 2019.
- [22] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 10, pp. 84648–84663, 2022.
- [23] R. D. Miller, "Joining two lines with a circular arc fillet," in *Graphics Gems III (IBM Version)*. San Mateo, CA, USA: Morgan Kaufmann, 1992, pp. 193–198.
- [24] J. Wen, X. Zhang, Q. Bi, Z. Pan, Y. Feng, J. Yuan, and Y. Fang, "MRPB 1.0: A unified benchmark for the evaluation of mobile robot local planning approaches," 2020, *arXiv:2011.00491*.



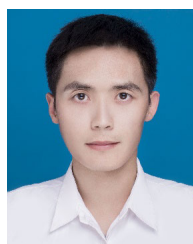
BINPENG WANG received the master's degree in control theory and control engineering from Jinan University, in 2004. He is currently an Associate Professor with the Shandong Academy of Sciences, Qilu University of Technology, Jinan, China. His current research interests include intelligent control, robot application, and motion planning.



DIANYUAN JU received the B.S. degree in automation from the Shandong Academy of Sciences, Qilu University of Technology, Jinan, China, in 2018, where he is currently pursuing the M.S. degree in control engineering. His research interests include path planning, motion planning, and particularly the motion planning of mobile manipulators.



FANGZHOU XU (Member, IEEE) received the master's degree in microelectronics and solid-state electronics from Shandong Normal University, in 2010, and the Ph.D. degree in communication and information system from Shandong University, in 2014. She is currently a Lecturer with the Shandong Academy of Sciences, Qilu University of Technology, Jinan, China. Her research interests include brain-computer interface and pattern recognition.



CHAO FENG received the Ph.D. degree in signal and information processing from Shandong University, in 2015. He is currently a Lecturer with the Shandong Academy of Sciences, Qilu University of Technology, Jinan, China. His current research interests include homomorphic encryption, remote data checking, and motion planning.



GUANGLONG XUN received the bachelor's degree in electronic information engineering from the Qilu University of Technology, in 2013. He is the Director of the Product Technology Center, Shandong Maiyue Wenbao Technology Company Ltd. During his work, he developed a number of products, such as pulp repair machine, spray deacidification machine, and immersion deacidification machine. He has published two invention patents, five utility model patents, and 11 software copyrights.