2018 IEEE 7th Data Driven Control and Learning Systems Conference
May 25-27, 2018, Enshi, Hubei Province, China

RRT based Path Planning for Autonomous Parking of Vehicle

Kaiyu Zheng, Shan Liu

College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China
E-mail: sliu@zju.edu.cn

Abstract: Path planning is one of the most issues in the automatic parking system for vehicle. This paper presents a path planning method based on rapidly-exploring random tree (RRT) with non-holonomic constraint and kinematics model of vehicle. First, the kinematics model of car parking according to the vehicle kinematics equation is set up, and the non-holonomic constraints are put forward. Based on this model, the RRT algorithm is used to search parking path with the constraints. And then, to optimize the search efficiency, two strategies--target preference and bi-RRT are used and also the cost function is added for optimization. Besides, because of the new detected obstacles, a replanning method is used to replan the path using the feature of the RRT algorithm. Finally, the performance of the proposed method is verified on a simulation model based on matlab.

Key Words: Automatic-parking path planning, non-holonomic constraint, RRT, replanning

1 Introduction

The problem of planning a path for parking an autonomous vehicle in narrow environment is considered in this paper. As there are more and more cars nowadays, the pressure on parking lots has exploded, and there are less and less parking space available. In general, the driver mainly relies on the rearview mirror to observe the rear of the car. However, because the rearview mirror is convex mirror, it will cause distortion of image, which will affect the driver's judgment. Besides, there is a large area behind the car, which cannot be seen through the rear-view mirror, and often brings huge hidden danger to parking. On the other hand, as the number of vehicles increases every year, the number of new drivers is increasing year by year, as well as the problems caused by unskilled driving. The increasingly crowded parking space have increased the tension accumulated during work, which affects the quality of life. Thus there is a desperate need for a quick solution to the parking problem, which is the automatic parking system.

The current automatic parking system is divided into two types, one is the parking auxiliary system and the other is the automatic parking system. The former uses ultrasonic sensors, cameras and radar equipment to detect the environment, to provide a broader field of vision for the driver, and to assist in parking, but the steering wheel, the brake and accelerator operation is performed by the driver manually. The latter is fully automatic control, including collecting environmental information through sensors and generating a parking path. The parking system automatically controls the steering wheel to complete the parking operation.

Path planning is a key problem to be considered in automatic parking system. At present, the path planning can be divided into two categories according to the acquisition way of surrounding environment information. One is the global path planning method based on known environmental information. The second is the local path planning method based on uncertain environmental information, such as vector field histogram method, polar vector field method and potential field method, etc. In the local path planning method, the environmental information is unknown or partially unknown, that is, the size, shape or position of the obstacle cannot be obtained directly through the sensors.

First of all, it is easy to think out that we can consider the path planning problem from geometrical aspect: calculate the minimum turning radius under the geometric constraints of the environment and according to the constraint of the minimum turning radius, a better path can be obtained through some geometric programming, but this path needs to be guaranteed to be collision free.

Typical studies are as follows: Dubins [1] proposed the shortest path planning method for vehicles using the minimum turning radius. The path of the vehicle is from any starting positions to any target positions. On the basis of Reeds and Shepp's research on the shortest path problem of vehicles with changing direction [2], Kanayama and Hartman [3] developed a smooth path planning method based on three curves. This method defines two cost functions for smoothing paths: path curvature and derivative of path curvature. These definitions are used to obtain two simple paths: arc and cubic spiral. It has been proved that the path curve obtained by this method is smoother than that of Clothoid Curves. The Brussel and Schutter [4] proposed a feasible path consisting of a straight line, an arc, and a spiral.

Generally, the parking method can be summarized into three types: parallel parking, vertical parking and ladder parking. In general, the above studies only focus on one or two of these three problems, lacking of generality, and the real-time calculation is more complex. Long Han put forward a path planning method [5] based on the bidirectional RRT (rapid search random tree) to deal with different situations, which is a more general way, and can be further extended to the road.

In this paper, we first analyze the nonholonomic constraints and build a kinetic model of the vehicle. And then we propose some extensions of RRT, which includes redefining the cost function and collision detection function to adapt to path planning for parking in the narrow

*This work is supported by the Public Welfare Technology Application Research Project of Zhejiang Province, China under Grant 2017C31047

environment. At last, the results of simulations are given and conclusions are drawn.

2 Kinetic Model with Nonholonomic Constraint

In the field of mobile robots (including vehicles), nonholonomic constraints are very common, and they are one of the limits to perfect motion planning. This section describes the nonholonomic constraints and carries out kinematics modeling for vehicles under the condition of nonholonomic constraints to lay a foundation for the subsequent path planning.

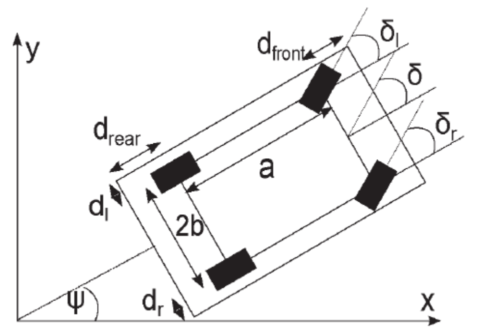

Fig. 1: Vehicle model and coordinate system

Assuming that the vehicle is a rigid body during the movement, its position space is three-dimensional, denoted as:

$$q = (x, y, \psi) \tag{1}$$

where x and y is the generalized coordinates, ψ represents the angle between the orientation of the vehicle and the frame. In addition, the center of the rear wheel of the vehicle is the origin of the body frame, and the distance between the front and rear axle is a, and the coaxial wheel spacing is $2b$.

Because the wheels can purely roll, the speed of the vehicle at any given moment is oriented towards the main axis of the vehicle. When Δt goes to 0, it satisfies the following differential constraints:

$$dy / dx = \tan \psi \tag{2}$$

Assuming that the speed of the car body is v, by solving it, it can be obtained:

$$\begin{cases} \dot{x} = v \sin \psi \\ \dot{y} = v \cos \psi \end{cases} \tag{3}$$

According to the geometric relationship, when the steering Angle is δ, instantaneous center of turning is C1:

$$\begin{cases} R_E = a / \tan \delta \\ d\psi = v / R_E \end{cases} \tag{4}$$

Thus, we can simplify this:

$$\dot{\psi} = v * \tan \delta / a \tag{5}$$

The control variables of the vehicle are respectively:

u_v : vehicle speed control input

u_δ : vehicle steering angle control input.

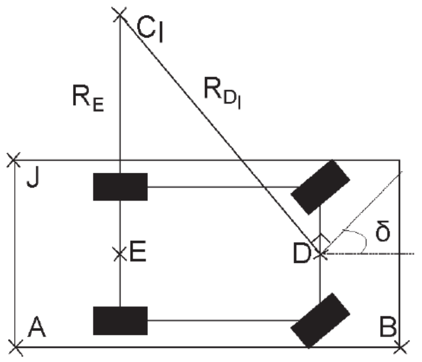

Fig. 2: Vehicle's instantaneous center of turning

To sum up, the kinematics model of the vehicle can be expressed as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ \tan\delta / a \\ 0 \end{bmatrix} u_v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_\delta \tag{6}$$

3 Extended RRT Path Planning Method

3.1 Bi-RRT

The fast RRT in this paper can meet the basic requirements of automatic parking path planning. Since it is a search method to cover the search tree map, it does not need to model the parking space in advance, and it has good adaptability to each type of berth. In addition, this method can easily add obstacle constraints when searching for new nodes.

The pseudocode is described below:

```
1  bi-RRT()
2  Tree_a←q_init, Tree_b←q_goal;
3  While !flag
4    flag=false;
5    if rand(1)<0.1
6      q_rand←q_goal
7    else
8      q_rand←GetRandomNode(Xobs, Xfree);
9     q_nearest←FindnearestNode(Tree, q_rand);
10    q_a, u_best←Grow_tree(q_nearest, q_rand);
11      if q_a
12        q_b, u_best←Grow_tree(q_nearest, q_a);
13        if q_b
14          Tree_a←Tree_a{q_a};
15          Tree_b←Tree_b{q_b};
16          distance_a=min(distance(Tree_b, q_a));
17          distance_b=min(distance(Tree_a, q_b));
18          if  min(distance_a, distance_b)<ebs
19            flag=true;
20            Return Tree_a, Tree_b;
21        Exchange(Tree_a, Tree_b);
```

By the method above, RRT with target bias first uses GetRandomNode () function to select a point called *q_rand* randomly within the set *Xfree* or select the target directly. And then FindnearestNode () function is uesd to find the nearest node *q_nearest* to the sampled point from the tree according to the distance function. Next the Grow_tree ()

function is used to generate *q_a*, and *tree b* will set *q_a* as its goal point to extend itself. Then when the distance between the new generated *q_a* and any nodes on the *tree b* is less than the maximum tolerance or the distance between new generated *q_b* with any nodes on the *tree a* is less than the maximum tolerance, a suitable path is found. Finally, exchange two trees. Thus one extension is over.

3.2 Distance Function

RRT algorithm is trying to connect a new node to the nearest node in the tree, which will minimize the computational expense of the collision detection. Because the probability of collisions with shorter path are smaller, so the algorithm can quickly cover sampling space and will not waste a lot of samples. In general, Euclidean distance is used to measure the degree of proximity between nodes, but this kind of measurement function is more suitable for a system with holonomic constraints.

For a vehicle system with nonholonomic constraints, this distance measurement function needs to be changed. We should not continue to use Euclidean distance but consider the motion law of the vehicle itself. For example, in order to reach a position which is parallel to the vehicle, but not far from the vehicle, car needs a big turn or to go backwards and forwards to adjust the position. Using Euclidean distance in this case is not logical. The most accurate representation of distance is the simulation path of the vehicle model, while accurate modeling will lead to heavy computational burden. It is also important to note that it is essential for parking problems to be taken into account in the direction of the body and the reverse motion of the vehicle. Of course, the steering angle of the vehicle should be abandoned, as the vehicle's final pose does not depend on steering. In distance indicators, the orientation angle of the vehicle should be an important factor. In order to get a better distance metric, and try to reduce the computational burden, some approximate methods are adopted here:

$$D(P,P_0)=\begin{cases}2\pi R, & if \|Q_1-P\|<R \ or \ \|Q_2-P\|<R \\ \|P_0-P\| & otherwise\end{cases} \tag{7}$$

where R is the steering radius of the maximum steering angle of the vehicle (that is, the minimum steering radius). Q_1 and Q_2 are the centers of the smallest circle on both side, the final distance measurement should be expressed as:

$$M(q_1,q_2)=\sqrt{D^2(P_1,P_2)+\alpha(\psi_1-\psi_2)} \tag{8}$$

$$q=(x,y,\psi,\delta)^T=(P,\psi,\delta)^T \tag{9}$$

where α is used to define the weight of the orientation angle.

This definition of distance measurement is reasonable. The vehicle needs to travel a long distance to reach the point in these two circles mentioned above; When the nodes that need to calculate distance are located outside the above regions, the Euclidean distance is appropriate for the measurement of distance.

3.3 Collision Detection Function

The use of collision detection function ensures that each RRT tree node is obstacle free, but it is not clear whether the collision happens or not on the way connecting the parent node and children nodes. This paper selects the appropriate control set *u_best* = (*velocity*, *steer_angle*) according to the vehicle kinetic model. Use the selected *u_best* to integrate angular velocity and linear velocity for the same time interval to obtain an arc path from parent node to children node. To ensure that the intermediate process does not collide with the obstacles, take same points evenly over this arc. Then at each point use obstacle detection function to check. If collided, this section of the path and *q_new* will be abandoned.

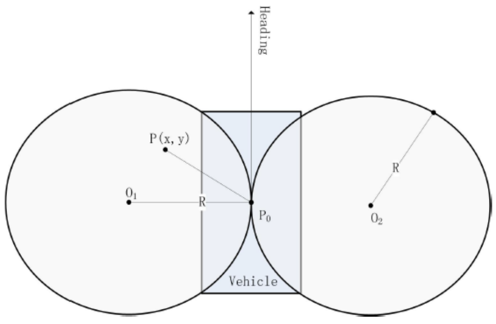

Fig. 3: Circle corresponding to the maximum steering angle

3.4 Cost Function of Path

When the RRT method finds a feasible path, there is no guarantee that this path is optimal or suboptimal, so the cost function of the path is introduced. In the set of all possible paths, compare the cost function of all nodes of each path and the current optimal path is given.

In addition, due to the parking problem, the distance cost is not the only indicator to be considered. The change of the vehicle orientation angle is also one of the indicators of cost function. When the field is empty, the limitation of obstacles is less. If not join the orientation angle variation into costs, the algorithm will return a path which is not in accordance with the usual practice of an actual person and is not conducive to vehicle control. The weight of the change of the orientation angle can be represented by the β.

The concrete implementation method is to record the path length and the change of orientation angle as two extra attributes at each node. After finding a feasible path, the two weighted sums of all points on this path are summed to obtain the cost:

$$cost=\sum l+\beta\sum\Delta\psi \tag{10}$$

3.5 Replan Method

The detection distance of the sensor system of the vehicle is limited, thus parking system can only get part of the environment information. With the movement of the vehicle, sensors will detect new obstacles, and the original built parking environmental information will change, include the obstacle constraints, which could make the currently executing optimal path infeasible. And then we need to replan the parking path.

RRT algorithms have advantages in replanning. Because RRT algorithm sample points in the environment, and the new sampled point is added to the tree to realize the search of the path. When the environment changes, the edge between some nodes will fail to meet the obstacles constraints, at this time we don't have to recreate a new tree. We can keep those that still meet obstacles constraints, and then cut off the edges and nodes that do not meet the constraints and continue to expand on the basis of the tree nodes, namely, replanning. This method can save a lot of time.

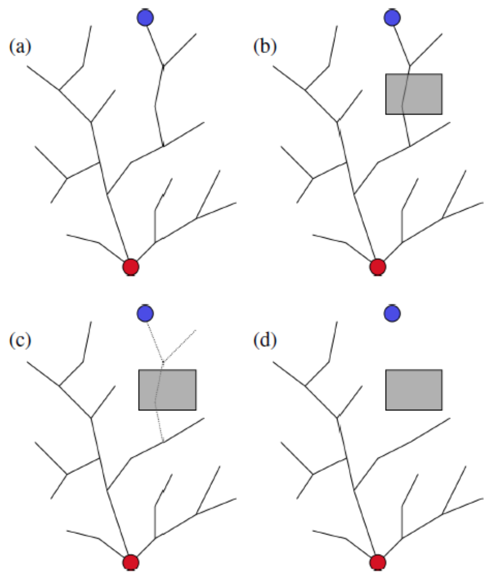

Fig. 4: Pruning process

As shown in figure 4, when the obstacle information is updated, the nodes that do not conform to the constraint need to be cropped. The approach is: when the edge between node A and node B collides with obstacles, assuming that A is B's parent, remove edge AB, B and children nodes of B from the RRT tree, to ensure the edges of the tree meeting the obstacle constraints. After updating the RRT tree, we can continue to plan a path. First check whether the previously found optimized path completely exists in RRT tree. If any, set the optimized path as the return of the algorithm. If it does not exist, continue to extend the RRT tree until a feasible path is found.
The pseudocode is as follows:

```
Replan
1  while SampleNum<MaxSampleNum
2     CheckMap();
3     if changed
4        CheckBranches();
5        TrimTree();
6        UpdateCandidatePath();
7     Grow_tree();
8     if find path
9        CandidatePath←CandidatePath ∪ {path};
10       path_best←Findthebest(CandidatePath);
11       show path_best;
```

In replan algorithm, first refresh the map. If changed, begin pruning operation, and at the same time according to each subtree retained in the trees, update the current set of feasible paths. If there are nodes in these paths that are removed from the tree because of pruning operation, these paths will be nullified. Then the normal grow tree operation is followed, and the RRT tree is extended to add the feasible path to the alternative path, select the optimal path, and return the path to the parking system.

4 Simulation

To test the validity of improved RRT algorithm on automatic parking, we test the algorithm for different garages, and observe adaptiveness of the algorithm facing different berths and the feasibility of the path.

The steering angle of the vehicle is recorded in the parking process. The vehicle is in constant speed, with a speed of 2m/s, and the length of the body is a=4.5m and width b=2.4m. In order to calculate conveniently, the map and vehicle dimensions are multiplied by 10 for simulation. The maximum tolerance for finding the path is set as 5. The weight of the Euler distance and angle deviation (radian) in the distance function is equal to 300, and the weight of the distance COST and angle change COST in the COST function is equal to 1.

The simulation test is carried out on PC of Inter(R) Core(TM) i5-3230m @2.60ghz 8GB, and the compilation environment is Matlab2014b.

The vehicle model is represented by a simple rectangular frame, and the main shaft is indicated by red rough lines, and the part of the main shaft extending the rectangular frame is the head of the vehicle. The blue line represents the branch of tree B, and the red line represents the branch of tree A.

4.1 Parallel Berth

The starting point is [180,60, PI], and the target point is [120,25, PI], and the ratio of coordinates to actual size is 10:1.

The first feasible path is found at the 62nd sampling.

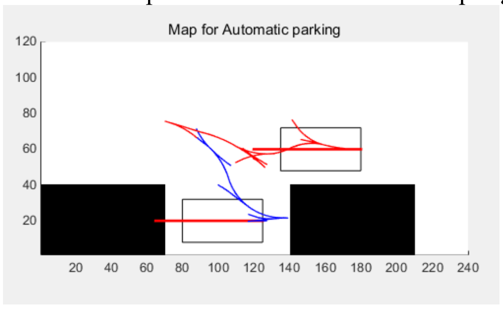

Fig. 5: RRT tree when find a feasible path

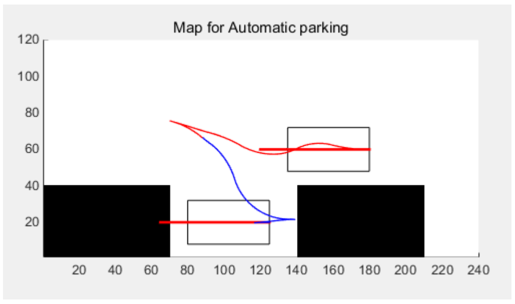

Fig. 6: a feasible path for parallel berth

4.2 Vertical Berth

The starting point is [200,120, PI] and the target point is [125,10,0.5 PI].

When sampling the 140th time, the appropriate path is found.

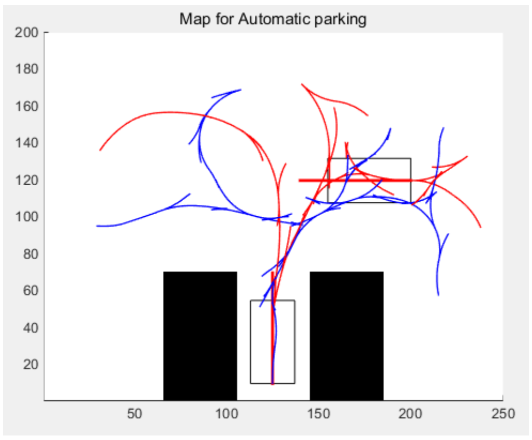

Fig. 7: RRT tree when find a feasible path

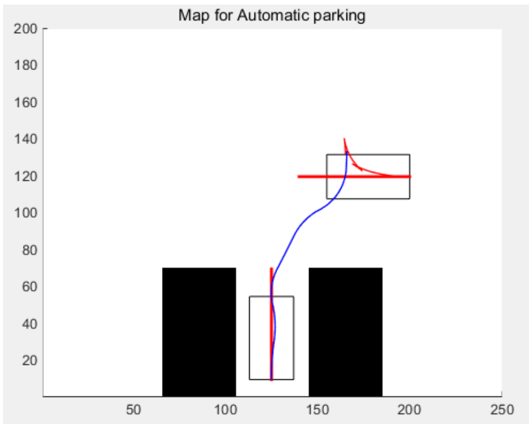

Fig. 8: a feasible path for vertical berth

4.3 Trapezoidal Berth

The starting point is [50,120,0], and the target point is [120,20, PI /4].

The first track found on the 102nd sampling is:

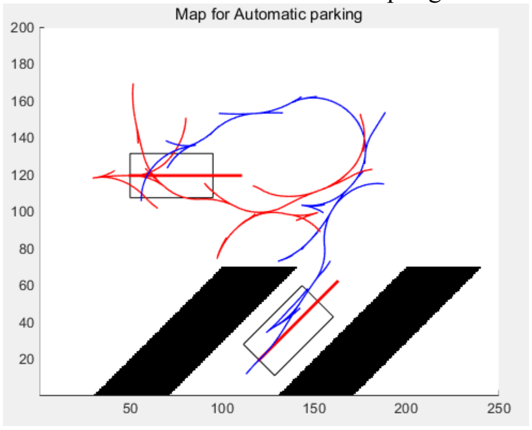

Fig. 9: RRT tree when find a feasible path

Results: The algorithm can quickly find a relatively optimal parking path, and in the subsequent search COST function (including path length and orientation angle variation) as the performance index, continue to optimize the generated path, to achieve the effect of gradual optimization.

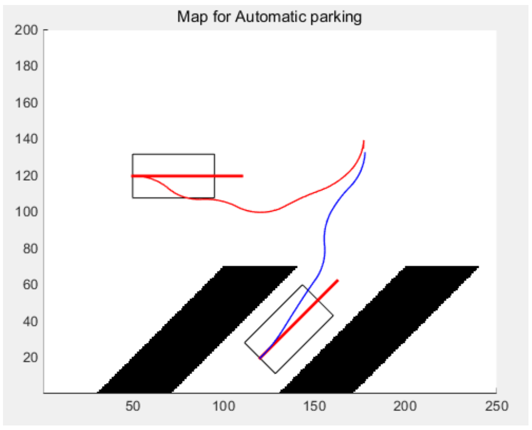

Fig. 10: a feasible path for trapezoidal berth

4.4 Replanning

A feasible path was found at the 38th sampling for the side parking space shown in figure 11.

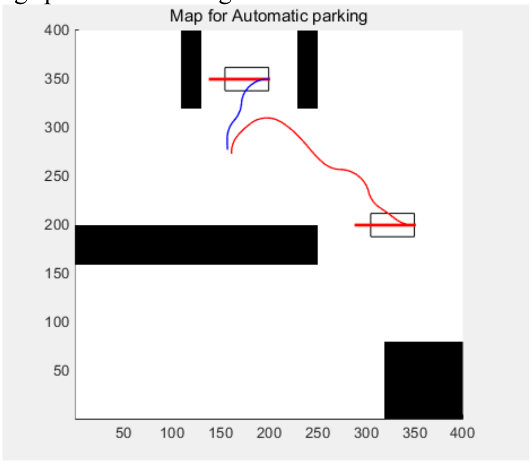

Fig. 11: a feasible path for side parking space

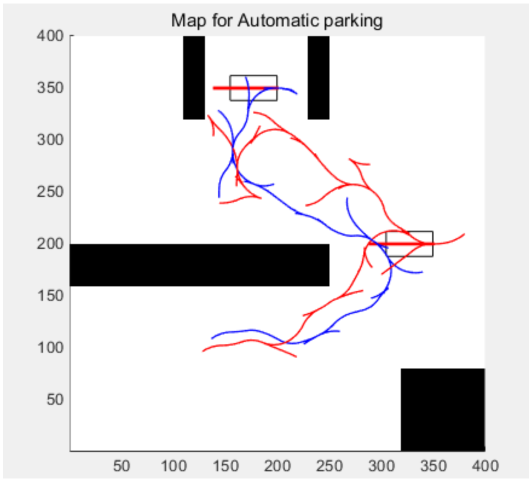

Fig. 12: RRT tree when find a feasible path

On the 100th sampling, add a newly detected obstacle. The newly detected obstacle is a square object with coordinates of [140,240] and the length of 20.

Because of the detection of new obstacle, the map needs to be updated and the tree has to been pruned.

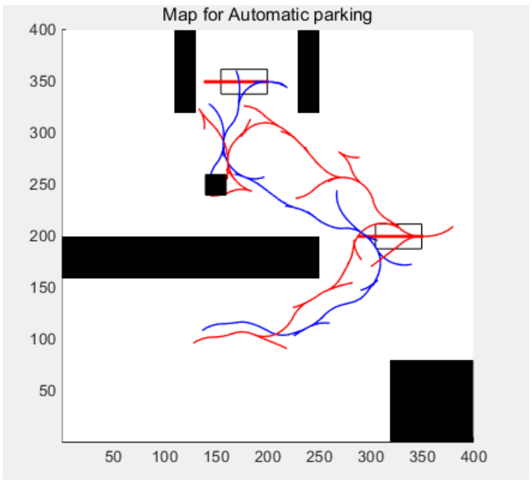

Fig. 13: RRT tree when add new obstacle

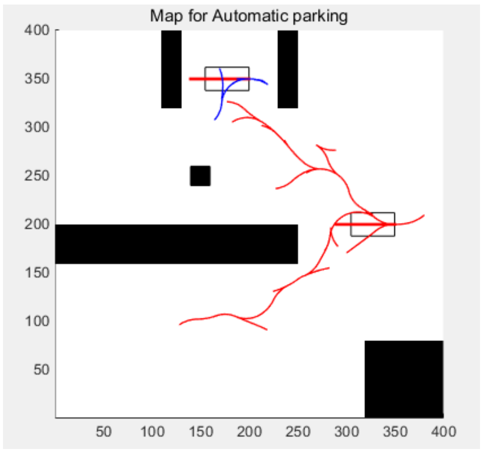

Fig. 14: RRT tree after pruned

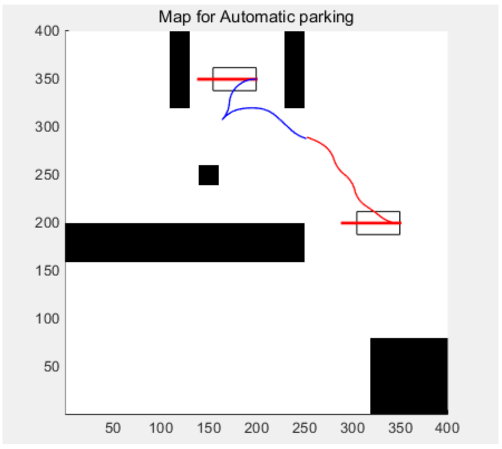

Fig. 15: a new found feasible path after tree prunning

According to the simulation results, replan algorithm can adjust the structure of the RRT tree when the environment changes without wasting the sampling points available for current environment. Compared with the former algorithm, this algorithm saves a lot of time.

5 Conclusions

For RRT algorithm, two fast methods are adopted to increase the speed of growth to target node. Moreover, the cost function of the original RRT algorithm is modified by introducing the orientation angle variations, which improve the performance of the algorithm. After considering the model of the robot motion, we make a rewritten cost function to reasonably determine the distance between each position, which results the path more realistic. In addition, the cost function is added so that the algorithm can quickly give a suitable path, and can continue to optimize the path with cost function.

For undetected obstacles, we put forward a thought of replan, update search tree and alternative paths, and replan, which is of great advantage compared with traditional geometric programming method.

References

[1] L. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal position and tangents, *American Journal of Mathematics*, 79(3):497–516, 1957.

[2] J. Reeds, and L. Shepp, Optimal path for a car that goes both forward and backward, *Pacific J. Mathematics*, 145(2): 367−393, 1990.

[3] Y. Kanayama, and B. Hartman, *Smooth local path planning for autonomous vehicles*. New York: Springer-Verlag, 1985, chapter 16.

[4] H. Brussel, and J. Schutter, Hierarchical control of free-navigation AGVs. Proc. Int. Workshop on *Information Processing in Autonomous Mobile Robots*, Munich, 105−119,1990.

[5] L. Han, Q. Do, and S. Mita, Unified Path Planner for Parking an Autonomous Vehicle based on RRT, in *IEEE International Conference on Robotics and Automation*, 2011

[6] X. Lan, S. Cairano, Continuous curvature path planning for semi-autonomous vehicle maneuvers using RRT, in *European Control Conference(ECC)*, 2015, 2360 – 2365.

[7] E. Dönmez, A. Kocamaz, and M. Dirik, Bi-RRT path extraction and curve fitting smooth with visual based configuration space mapping, in *International Artificial Intelligence and Data Processing Symposium (IDAP)*,2017,1 – 5.