Path Planning for Autonomous Vehicles in Complicated Environments

Xingxing Du,Xiaohui Li,Daxue Liu,Bin Dai
College of Mechatronic Engineering and Automation
National University of Defense Technology
Changsha, P.R.China
dxxing43@163.com

***Abstract*—When the autonomous vehicle is running in complex unstructured road and the reference path to guide the vehicle could not be obtained directly, at this time, only sparse task list point left as the reference information. To generate a safe and smooth path in this situation for autonomous vehicle to run, we propose an improved heuristic graph search path planning algorithm to solve the problem by considering both the obstacle constraint and the vehicle model. To ensure the planning result is executable, kinematic constraints of the vehicle is considered when designing the motion primitives; for the efficiency and safety, obstacle constraint and the vehicle motion were taken into account when designing the cost and heuristic functions; in order to ensure that the vehicle could accurately reach the target pose, Reeds-Shepp curves are used to connect the target pose, and numeric non-linear optimization method was used to improve the smoothness of the plan result. Experimental results in the autonomous vehicle show that the methods in this paper is effective.**

***Keywords*—path planning; heuristic; reed-shepp; optimization;**

I. Introduction

While the vehicle is running in a partially unknown or completely unknown environment, such as the parking problem on parking lot. At this time, the Reference path could not be directly obtained. And lacking of reference information could result in calculating a continuous and collision free path complicate. Some kinds of sampling algorithms could generate local path that can handle these complex obstacle avoidance problem, but they require a reference path for guidance, so it does not apply to solve this problem. A way to solve the problem is to use the sampling motion planning method based on control space, this kind of methods use reactive obstacle avoidance strategy which could guarantee the generated trajectory advisable and ensure the safety of the vehicle. However, the length of generated path is usually too short with these methods and the trajectory generation propose shows no consideration of the target pose constraints, which makes the vehicle fall into local minima easily. When the external environment constraint is strong to the vehicle attitude, these methods could not guide the vehicle to reach target attitude accurately. Another idea is to converse the original problem into an optimization problem in continuous state and control space. However, due to the complexity of the vehicle and environment constraints, the convergence of the optimization problem is difficult to guarantee.

For the actual autonomous vehicle motion planning problem, in addition to the needs of considering the optimality of the solution and the completeness of algorithm, other two important factors should also be considered are: the real-time performance of the planning algorithm and the smoothness of

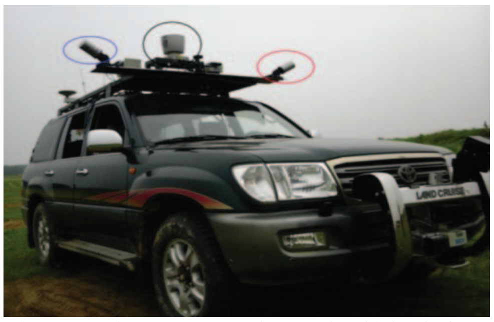


Fig. 1 Experiment Unmanned Ground Vehicle Platform

the output trajectory (planning result). In order to improve the real-time performance , improved heuristic function was used for the search graph construction ,and succeed to reduce the number of nodes expanded and can also improve the efficiency of search algorithm; to improve the smoothness properties of the output trajectory, an proposed optimization method based on gradient descent is used to optimize the original path planning results .The planning results of Search algorithms based on graph search method are combination of trace element series. Due to the curvature mutation of adjacent motion primitives in connection points, the curvature of planning result is not continuous, thereby, the lateral tracking control performance will be affect. Therefore, the improvement of original planning results could effectively improve the smoothness of the lateral control.

II. Related Work

In order to solve the problem of motion planning in complex obstacle environment, many scholars discrete the state space of the autonomous vehicle and then the motion planning problem is formalized as a multi-step sequence decision problem [1,2,3,4,5]. Specifically: first, use the state sampling strategies to construct the search graph and take the sampling state as the vertices of the graph, then use graph search methods to extend the graph edges until connecting the start vertex and the target vertex. This kind of methods based on state sampling graph search algorithms and can be divided into two parts: the graph search method with deterministic sampling and the graph search method use random sampling. Based on the random sampling graph search algorithm, we can quickly get a feasible solution but the planning results are usually suboptimal and the solution quality usually directly depends on the heuristic function which is used to guide the search. This kind of methods are mainly used for solving complex high dimensional motion planning problem [6,7,8,9]. In comparison, deterministic sampling graph search method can guarantee the optimization and the completeness in a specific resolution, in next parts we use idea based on the heuristic graph search to solve motion planning problem for autonomous vehicle in wide range environments.

III. HEURISTIC GRAPH SEARCH METHOD

Here we assume that target state of planning could be obtained, and that the information of local environment around the vehicle with the vehicle position, attitude could be updated in real time through the sensing system of the vehicle. The planning task is based on the input information, and the planning goal is to calculate an obstacle avoidance and executable path from the current vehicle attitude to reach the target attitude.

To guarantee the optimality of the solution, we use the heuristic graph search algorithm to solve the motion planning problem [10]. The basic idea of heuristic graph search algorithm is when in the process of constructing search trees, put extended node into the Closed list, and insert the node which has not been extended into the Open list. In the graph search process, use the cost function $f(s)=g(s)+h(s)$ to assess the cost of each node in the search tree to determine the order to expand them .$g(s)$ is the actual distance cost(usually the path length) from the start node to current node s, $h(s)$ is evaluation of the cost from current node s to the target node , also known as the heuristic cost.

A. Environment Representation and Search Graph Construction

We mainly study the motion planning problem in the two-dimensional plane, and take the external environment as Grid map, which could effectively simplify the environment. Traditional graph search method usually use 4-connection or 8-connection method to extend node and build the search graph G (V, E),which takes the center of each original obstacle gird in the grid map as a vertex V in graph and take segmented linear trajectories as edges E in the graph search, all vertices locate at the center of the grid. Some Scholars improved it, making the vertices enable to locate on an arbitrary point of four sides of the grid[11]. However, due to the non-holonomic constraints of the vehicle, the planning result is hard to be executed. In order to ensure the executable property of the planning results, here we use the motion primitives to build a search map, as shown in Fig.2 (b).

The State Grid is a representation of the discrete control space of the autonomous vehicle. Vertexes of state grid are discrete vehicle states, and the edges are extension of each vertex.

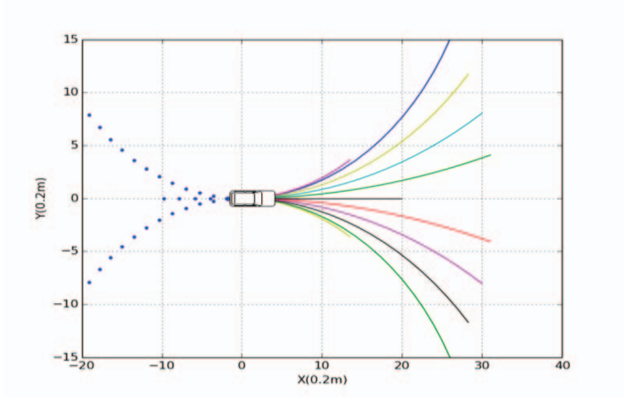


Fig. 2 Basic set of motion primitives

From Fig. 2 we can see that the end of motion primitives in state grid could be any point in the obstacle grid instead of being confined to the center or edges of the grid .When designing the motion primitives ,we considered the model constraints, so we used the State Grid to construct search graph instead of using 4 or 8 - connections to ensure that the planning result is continuous and executable .To keep the car more likely to go with straight line , we gives every motion primitive an additional cost coefficient , more curved the primitive is , higher cost it has. In Fig. 2, the shown motion primitives are the smallest motion primitive set for the search algorithm, the length and the number of motion primitives depends on resolution of the State Grid, and it will directly affect the efficiency of the planning algorithm. If we do not consider the real-time performance and computation efficiency of the planning, by increasing state grid resolution (i.e. to improve element numbers in minimum primitives set, and reduce each length of each motion primitives), the scope of solution space will be increased, which would improve the ability to solute motion planning problem under complex environment, but it would also consume a large amount of computing resources. By the obstacles grid map resolution limit, if the length of motion primitives is much smaller than the length of obstacle grid's edges, then, it will consume more computing resources and will not improve the solution performance for planning. From the practical point of view, we compromise on real-time computing requirements and state grid resolution. Our Motion primitives includes the long primitives and the short primitives ,the maximum curvature of motion primitives meet the vehicle minimum turning radius constraint, long primitives' forward expand speed is faster, and short primitives are more flexible to deal with complex obstacles , we also designed backing primitives contains the left-right three primitives and meets the same vehicle turning radius constraint.

Based on the State Grid, we use four-dimensional $X(x, y, \theta, d)$ to represent the vehicle state, (x, y) is the position of vehicle, θ is heading of the vehicle, and we use 24 discrete values to represent all directions. d is a binary value represents the vehicle movement direction, 0 for backward movement and 1 for forward movement . On each individual motion primitive element, all the d value remains unchanged, the purpose of the backing primitives is when the vehicle is running in complex obstacles environment, sometimes we need reverse actions to get out of obstacles. In practical applications, we try to reduce the vehicle to carry out the reverse primitive, so we set a much higher additional cost for backing primitives. The four-dimensional states we use could ensure the continuity of the planning results. Here we do not directly consider trajectory curvature state, because the consideration of trajectory curvature state will be a significant increase in difficulty of designing motion primitives. In order to improve the smoothness of the planning results, nonlinear optimization method were used to smooth planning results in the post process .And velocity planning is based on the path planning result.

B. *Heuristic Function Design*

Influence of the heuristic function to A* algorithm is very large, if the heuristic function is too underestimate of the optimal cost ,the algorithm will increase the number of nodes expanded to a great extent. The traditional A* algorithm based on Euclidean distance or Manhattan distance to design heuristic function, although this heuristic function is admissible , but they ignore the vehicle kinematic constraints and obstacles constraints, which resulting in underestimated the cost of the actual trajectory in many cases.

Through the analysis, it is easy to find that the complexity of motion planning problem mainly comes from two aspects: one is the environment obstacle constraint; another is the vehicle motion constraint [1,2]. To ensure that the estimated cost of the optimal path is close to the actual optimal path cost, these two factors should be considered while designing the heuristic function. But consider these two factors at the same time will increase the difficulty of the design of the heuristic function. Therefore, we will consider it separately. We designed two kinds of the heuristic function h1(s) and h2(s), we use heuristic h1(s) for almost every node, except when the expanding node nears the target pose, we use heuristic h2(s) instead of h1(s). The designing process of the two kinds of heuristic functions will be analyzed in detail.

For the design of the heuristic function h1(s), we do not care about the direction of each node in graph, just ignore the kinematic constraints of the vehicle and considering only the obstacles constraint. In the obstacle two-dimensional grid with 8-connections, we calculate the shortest and collision free path from the starting point to target point, as shown in Fig.3 (a). Based on this way we can get the heuristic function which meets the admissibility requirements, and the cost is closer to the actual optimal cost compared with the Euler distance (ignoring the obstacle constraints). Here we use Dijkstra graph search algorithm to finish it. First, we construct a search graph based on the two-dimensional grid with 8-connections relations, and each grid is a vertex in the graph search, and set grid contains obstacles an infinite cost, then run the Dijkstra algorithm to calculate the shortest and collide free path for every grid in search graph to reach target point. At last, set the shortest path length as a heuristic cost function, such as shown in Fig.3(b).

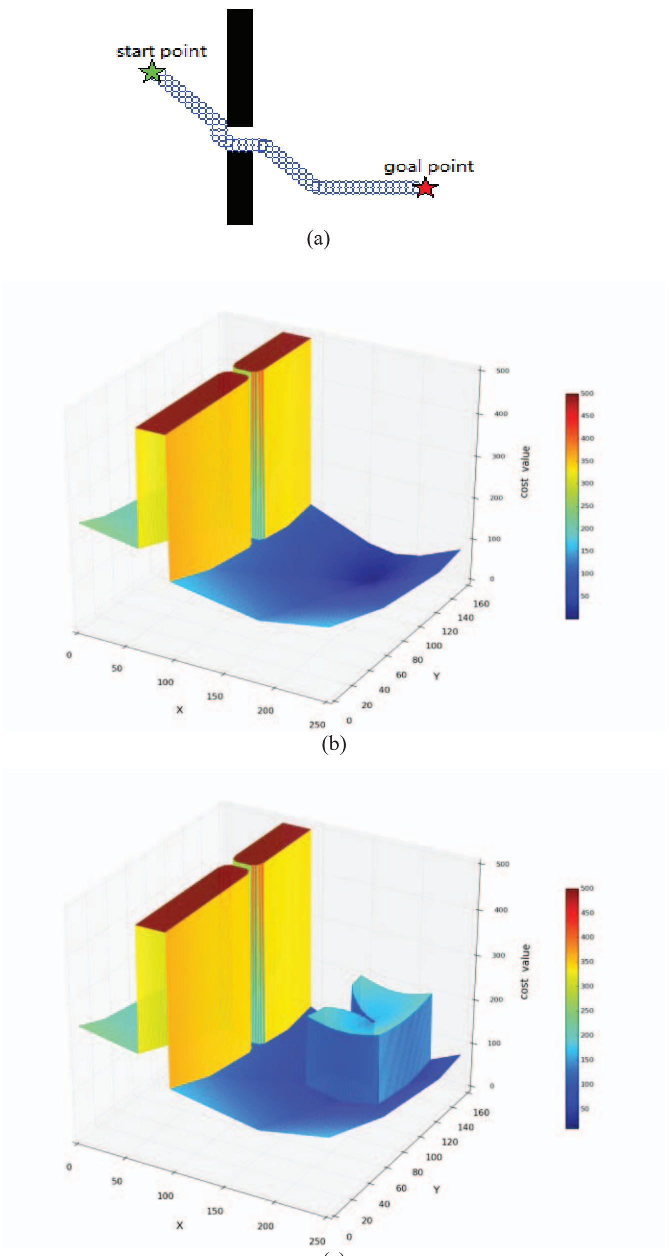
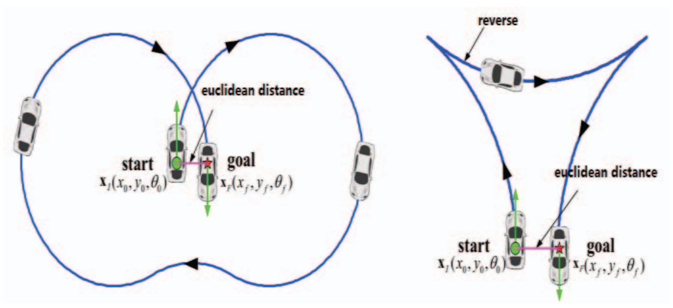


Fig. 3 Sketch of heuristic design: (a) The shortest path from the start point to the target point (b) heuristic function h1(s) (c) heuristic function h1(s) with h2(s)


euclidean distance
start
$\mathbf{x}_i(x_0,y_0,\theta_0)$
goal
$\mathbf{x}_f(x_f,y_f,\theta_f)$
reverse
euclidean distance
start
$\mathbf{x}_i(x_0,y_0,\theta_0)$
goal
$\mathbf{x}_f(x_f,y_f,\theta_f)$
Fig. 4 Dubins curve and Reed Shepp curve

As for the design of heuristic function h2(s), we consider the vehicle kinematics constraint. As shown in Fig.4, when calculating a path connect initial state and target state , when use the Euler distance as the heuristic function, the heuristic function value will be very small because due to the vehicle kinematic constraints, the vehicle could not directly translate from the start attitude to the target vehicle attitude. Left curve in Fig.4 is the famous Dubins curve[12], the curve uses arcs and line segments and connect them together to generate a curve which meets the start and target position constraint and heading attitude constraint. And the arc segment can also directly consider the maximum curvature constraint of vehicle (minimum turning radius constraint). The curve can also ensure that the length of the generated path is shortest; in contrast, right curve in Fig.4 is the Reeds-Shepp curve[13], it also keeps all the advantages that Dubins curve has, but compared with Dubins curve, it considers reversing trajectory as shown. Reed-Shepp curves is more suitable to be used in more situations than Dubins curve, but it will take more computation resources. When we compute the Dubins or Reeds-Shepp curve from a vertex to the target attitude , if the distance between the vertex and the target is very large ,the computed curve is very likely to crash with obstacles, at this time ,the use of Dubins or Reed-Shepp curve length as the heuristic function value will be useless compared with the h1(s) heuristic function , so we only use Dubins or Reed-Shepp curve length as the heuristic function when the current expanding vertex nears the target point. We pre-calculate many Dubins curves offline from vertexes of different positions and different headings to the fixed target attitude(target attitude set to 0 degree) to be the heuristic function. According to the actual situation to use h2(s), by translation and rotation, we could get the actual length for every vertex with different headings target pose. As shown in Fig.3(c),it can be seen that we use h1(s) as the heuristic function value when far away from the target ,and when nears the target pose, use the h2(s) as heuristic function .The h2(s) heuristic value shown in Fig.3(c) is just for show, it only contains one direction of vertexes near the target, in fact ,we have 24 discrete directions of nodes in each grid.

In the motion planning experiments we find that when obstacles are density in environment, by using the heuristic function h1 (s) combine h2(s) could significantly reduce the number of nodes expanded in planning process and improve the planning efficiency. Fig.5 shows the comparison between the motion planning results based on the Euclidean distance as the heuristic function (Fig.5(a)) and the h1(s) as the heuristic function(Fig.5(b)). Fig.5(c) shows the heuristic function value h1(s) combine with h2(s). The grid resolution in the experiment is 20cm× 20cm, (since there are no special circumstances, the obstacle grid map resolution in all planning experiment are 20cm× 20cm). Fig.5(a) is based on Euler distance heuristic function. From the figure we can see that in forward search process, the vehicle is always trying to connect the target attitude with the shortest straight line segments, but because the obstacle constraints, resulting in searching efficiency very low when using Euclidean distance as the heuristic function, it expanded more than 100000 nodes. In Fig.5(c),we use the combined h1(s) and h2(s) heuristic function , the expanded node number is 12113,compared with the Fig.5(b) where we only use h1(s) ,the reduced the number of expanded nodes is 1000. We can see that use h1(s) with h2(s) heuristic function could improve the compute efficiency a lot.

In the experiment we find that if we use motion primitives shown in the Fig.2 to expand nodes, when the expanding node is close to the target attitude, due to finite number of element in motion primitives set, it is difficult to guarantee the end of the generated trajectory precisely meet the target pose constraints. As shown in Fig.5 (b) and (c), when the trajectory is extended to near the target pose, because they could not get to meet the target pose constraints of trajectory, leading to the expansion of a large number of nodes. In order to overcome this problem, we use Reeds-Shepp curve to connect the current expanding node with the target attitude , then we test whether the generated curve will collision with obstacles , if collision happens , then continue to use the heuristic function we mentioned(h1(s) combined with h2(s)) to expand the node ,and

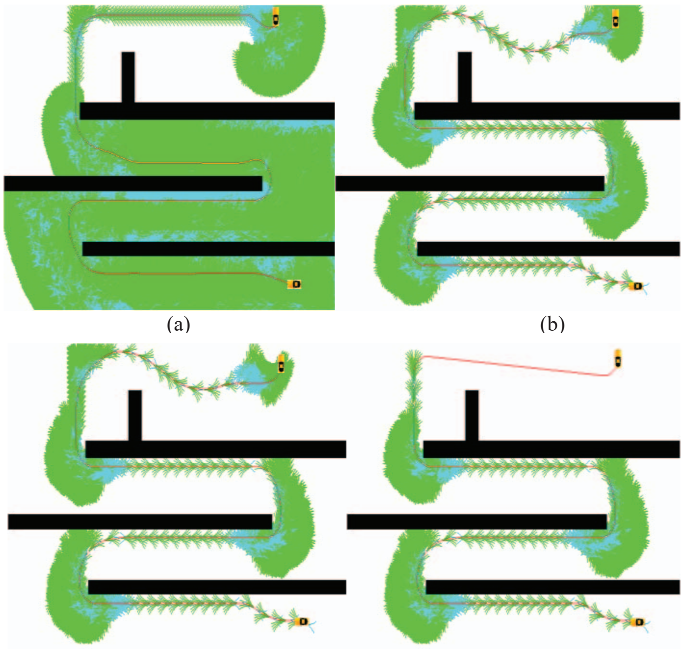


Fig. 5 Plannig results use different heuristic functions: (a) Euclidean distance (b) H1(s)function (c) H1(s) with H2(s) function (d)Reed-Shepp curve connect target

in next expansion period ,we try the Reeds-Shepp curve to connect current expanding node and target pose again; otherwise, stop the search and return the succeed path planning results. As shown in the Fig.5(d), the curve nears the goal is an Reeds-Shepp curve which directly connected to the target pose .In the experiments, we set the Reeds-Shepp curve radius constraint 5m correspond to fact. According to these experiments, we found that the number of extended nodes can be greatly reduced by this method, especially when the target attitude is located in the region of strong obstacle constraint. As shown in Fig5(d),the expanded nodes number is 11034.We use the Reeds-Shepp curve when the expanding node is close to the target pose, and the shorter distance they are, the higher probability to use Reeds-Shepp curve.

C. Path Optimization

In order to overcome the curvature discontinuity problem of the result paths, we post process the original path plan results through nonlinear optimization algorithm. Here we use nonlinear methods-conjugate gradient descent optimization method (conjugate gradient method) to process the path [79]. In the smooth process we also considered the left and the right road boundary constraints .Suppose the line below is our planning path $X_i(x_i, y_i)\ (0, 1, 2, \dots N-1)$.

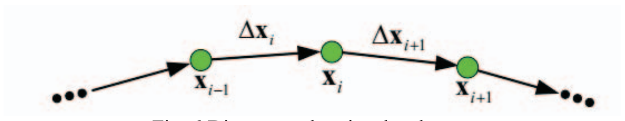


Fig. 6 Discrete path point sketch map

$\Delta X_i = X_i - X_{i-1}, (1 \le i \le N-1)$ is vector from X_{i-1} to X_i ,keep the start point and end point in the curve stable ,adjust other N-2 points in the middle. The goal is to minimize the function J:

$$J = \omega_1 J_1 + \omega_2 J_2 + \omega_3 J_3 \qquad (1)$$

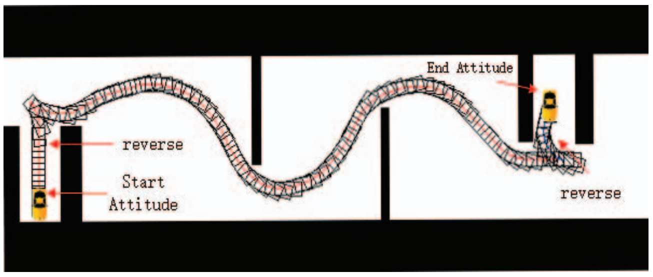
$$J_1 = \sum_{i=1}^{N-2} k_i^2, k_i = \frac{\Delta\phi_i}{|\Delta x_i|}, \Delta\phi_i = \cos^{-1}\left(\frac{\Delta x_{i+1}^T \Delta x_i}{|\Delta x_{i+1}||\Delta x_i|}\right) \qquad (2)$$

$$J_2 = \sum_{i=1}^{N-2} |\Delta x_{i+1} - \Delta x_i|^2 \qquad (3)$$

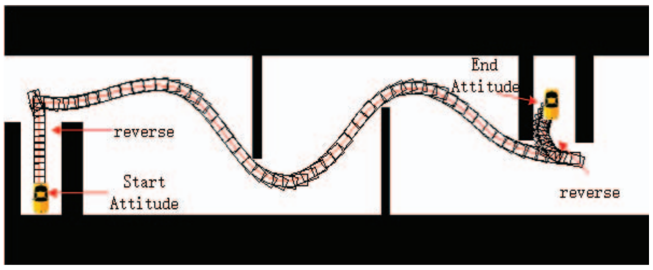
$$J_3 = \sum_{i=1}^{N-2} f(x_i), f(x_i) = ae^{b|x_i - x_{oi}|^2} + ce^{d|x_i - x_{oi}|^2} \qquad (4)$$

ω_1, ω_2 and ω_3 are the weight for penalty term. As we can see from above, *J₁* is the penalty of the path curvature, it is integral sum of the curvature square for the whole curve. *J₂* is a penalty for the direction change of the neighboring vectors, which is the integral sum of the variation of the neighboring vectors for the whole curve. The purpose of the above two penalty items is to minimize the curvature of the curve under the premise of ensuring smooth curve. Purpose of introduction of *J₃* is to consider the hard constraint of road boundary and ensuring that smoothed path is located between the left and right boundaries. Here we design a continuous differentiable nonlinear function *f(xᵢ)*. When the adjusted path is located in the inner boundary, *f(xᵢ)* value is small, and vice versa, once adjusted way is beyond the boundary, *f(xᵢ)* value will be very large. The unknown parameters a, b, c, d in *J₃* can be obtained based on the road width constraints or by artificial setting.

In the practice, we usually do not optimize the backing trajectory and the Reeds-Shepp curve segment. We only optimize the forward direction segment. The curve in Fig.7(a) is the original planning result, as shown , curve near the target attitude is generated by Reeds-Shepp, we do not the optimize it, and we also do not optimize the backing segment near start attitude. In Fig.7(b), we can see that after optimization, path has been smoothed ,the curvature comparison between the two path is shown in Fig.8. The optimized path is smoother than the original one.

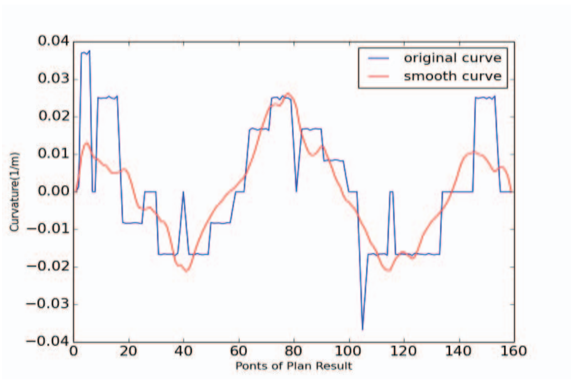


（a）



（b）

Fig. 7 Comparison of planning path before and after optimization: (a) before optimization (b) after optimization



a

Fig. 8 Contrast of curvature before and after optimization

In experiments, we also find that the smoothed path has certain deviation from the original collision free path, and this deviation may also lead the optimized path to collide with obstacles. Taking into account this problem, we used the iterative method to solve it. We adjust the penalty term parameters in formula (1), which could adjust the optimized path from the original path between the maximum lateral deviation path, so we could guarantee that optimized path not collide with obstacles. The specific process is as follows: if the collision happens, then reduce the maximum lateral deviation between the optimized path and the original path until there is no collision. The extreme case is that the lateral deviation of the hard constraints tends to be zero, then the smoothed path is same as the original path.

IV EXPERIMENT AND ANALYSIS

This part is about path planning experiments in actual obstacle environments which could verify the effectiveness of the algorithm.

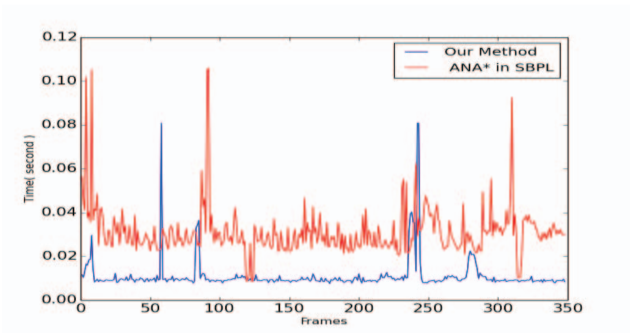


Fig. 9 Comparison of time cost with our method or ana* in SBPL

In order to detect the real-time performance of the algorithm, we used the sensor data collected by the laboratory autonomous vehicle platform to compare the method of this paper and the ANA* algorithm in the SBPL library. The obstacles map size in the experiment is 40 meters front and 10 meters behind the vehicle, left and right side is 25 meters beyond the vehicle. The map resolution is 20cm× 20cm, so the map has of 250 × 250 grids. The computer in all experiment is 4G ram with i5 4300u CPU, and operating system is Ubuntu 14.04. The initial heuristic coefficient in ANA* algorithm is set to 1 so that the ANA* algorithm likes A* algorithm. We run the two method with the same 350 frames sensor data collected by autonomous vehicle platform, the time cost of planning in every frame is shown in Fig.9, it can be seen from the chart that our method could improve the efficiency a lot in most cases.

We further tested out method in vehicle experiments with the autonomous vehicle platform. The map in the vehicle experiment is 20 meters behind the vehicle and 100 meters ahead of the vehicle, left and right side is 25 meters. The first scene is continuous winding in narrow path, the second scene is a simulated parking lot. In practice, our autonomous vehicle platform usually obtain accurate sensor data of obstacles around the 50 meter range take the car as the center , and beyond that range, recognition accuracy would decrease much , so we assumed that there is no obstacles beyond the range of 50 meters when we plan. In the vehicle experiment, we set a distant target point as the final target point ,and upon the search process, when the target point is 50 meters distance away from the current vehicle position , do not search the part out of the 50 meter range but just use Reed-Shepp curve connect to the target pose directly instead. In the process of dynamic plan, as only the local information could be obtained, so we use previous path when planning. Firstly, plan a path based on current sensor data and current vehicle attitude and target attitude, and then save the planning results, in next frame that the sensor information is updated, we test whether the preserved path collides with obstacles, if collides, then find a suitable point in the preserved path and take it as a new start point , replan a path from this point to the target point. If no collision, we continue to use the previous path.

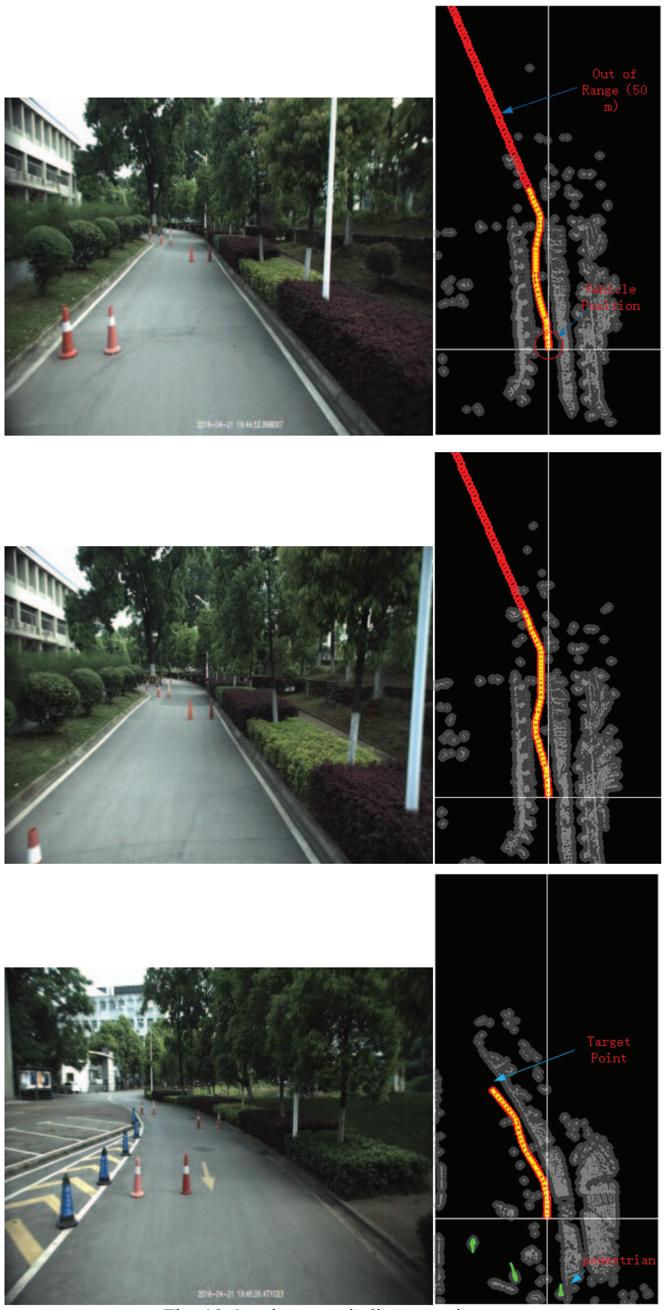


Fig. 10 Continuous winding experiment

The experiment results is in Fig.10.We took the experiment in campus. In the experiment goal attitude was preset (The Target Point in the figure), the vehicle start from a start point to achieve point ,and the vehicle can only get clear environment information not to far away ,it can not get the clear environment near the goal attitude. From the pictures we can see the path near the vehicle keeps consistent after replan, and distant path has been adjusted according to the new sensing information.

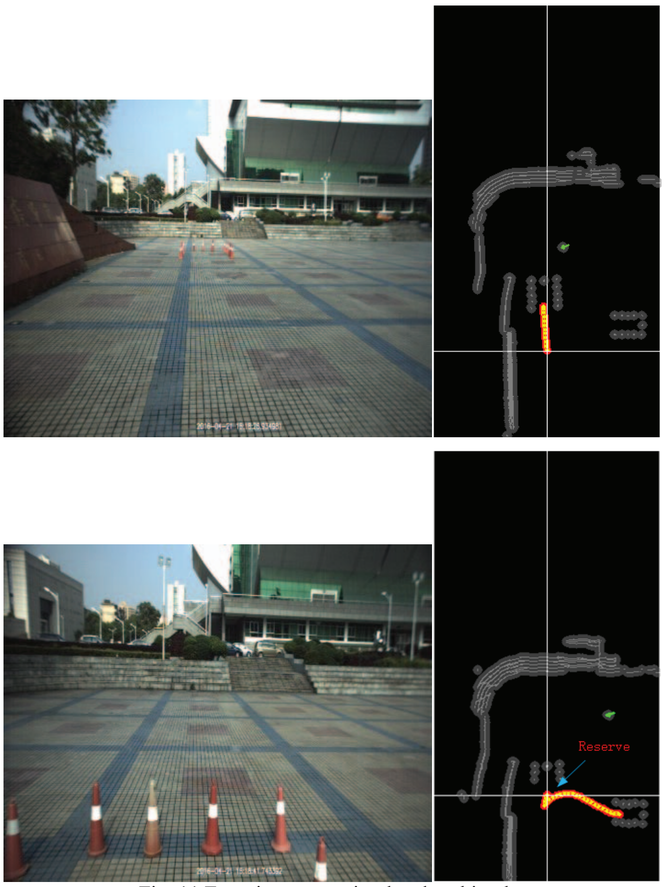


Fig. 11 Experiments on simulated parking lot

Experiments in the second scene is shown above. By pre-define multi target attitudes(the first target is in the upper parking space ,the second target is in the right parking space),the vehicle run from the start attitude to the upper parking space and then run from the upper parking space to the right parking space. The vehicle could continuous execute multi planning mission. At the same time, because the base motion primitives contains reverse segment, so as shown in Fig.11, the vehicle could perform the task of reversing. The experimental results show that this method can deal with various kinds of complex scenes.

V CONCLUSION

This paper introduces an improved algorithm to generate a path for autonomous vehicle, experiments show that the method is efficient and even can deal with parking lot misions. And our next goal is to change this method to deal with dynamic obstacles (pedestrians or cars) in more complex dynamic environments.

REFERENCES

[1] Bischofberger O. Planning Long Dynamically-Feasible Maneuvers for Autonomous Vehicles[J]. International Journal of Robotics Research, 2009, 28(8):214-221.

[2] Dolgov D, Thrun S, Montemerlo M, et al. Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments[J]. International Journal of Robotics Research, 2010, 29(5):485-501.

[3] Likhachev M, Gordon G J, Thrun S. ARA*: Anytime A* with Provable Bounds on Sub-Optimality.[C]// 2003:767-774.

[4] Koenig S, Likhachev M. Improved fast replanning for robot navigation in unknown terrain[C]// IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA. IEEE, 2002:968--975.

[5] Likhachev M, Ferguson D I, Gordon G J, et al. Anytime Dynamic A*: An Anytime, Replanning Algorithm.[J]. In ICAPS, 2005:262-271.

[6] Lavalle S M, Kuffner J J. Randomized kinodynamic planning[J]. International Journal of Robotics Research, 1999, 1(5):473-479 vol.1.

[7] Kavraki L, Svestka P, Latombe J, et al. Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces[J]. IEEE Transactions on Robotics & Automation, 1996, 12(4):566-580.

[8] Karaman S, Frazzoli E. Sampling-based algorithms for optimal motion planning[J]. International Journal of Robotics Research, 2011, 30(7):846-894.

[9] Kuffner J J, Lavalle S M. Space-Filling Trees: A New Perspective On Incremental Search For Motion Planning[C]// Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. 2011:2199-2206.

[10] Hart P E, Nilsson N J, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths[J]. IEEE Transactions on Systems Science & Cybernetics, 1968, 4(2):100-107.

[11] Ferguson D, Stentz A. Field D*: An Interpolation-Based Path Planner and Replanner[J]. Springer Tracts in Advanced Robotics, 2007, 28:239-253.

[12] Dubins L E. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents[J]. Rr N Boissonnat Czyzowicz Devillers Robert & Yvinec, 1957, 57(3):497-516.

[13] Reeds J A, Shepp L A. Optimal paths for a car that goes both forwards and backwards.[J]. Pacific Journal of Mathematics, 1990, 145(2):367-393.