2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Smooth Path Planning for Autonomous Parking System

Yang Yi*,[1], *Member, IEEE,* Zhang Lu[1], Qu Xin[1], Lei Jinzhou[1], Li Yijin[2] and Wang Jianhang[1]

***Abstract*— In this paper, we present a path planning algorithm for autonomous parking system. We focus on the kinematics of the car-like vehicle and improve the conventional geometric parking algorithm by proposing a new curve element named linearly steering spiral. A path planning algorithm based on smooth path searching and optimizing are presented. This method can generate smooth paths incrementally, and the reference control signals can be deduced directly once the path is determined. A simple closed-loop controller is designed in order to deal with the uncertainty from various aspects. Simulations are implemented in different scenarios including obstacle-free and cluttered environments, moreover, a real-world online experiment is executed. The results indicate that the proposed method achieves good performance on both accuracy and computational cost.**

I. Introduction

In the face of great traffic congestions from the increasing car parc in urban areas, which not only tremendously reduce the operation efficiency of our cities but also greatly increase the consumption of the fossil fuels, it is regarded that self-driving is an effective approach to compensate. As a self-driving behavior in specific conditions, autonomous parking has drawn much attention from researchers all around the world.

For path planning problem in autonomous parking, many methods have been presented in the literature. Considering both nonholonomic constraints and path continuity, methods based on reference functions achieve good performance. Iterative algorithms based on sinusoidal reference function [1] and quintic polynomial [2] can generate applicable paths but the strong dependency on parameters makes it difficult to apply these methods in practical conditions. This dilemma can be partly overcome by integrating fuzzy logic in the procedure of coefficients derivation [3]. Other methods based on splines such as η^3-splines [4] are convenient to generate smooth paths, however, it is difficult to take obstacles into consideration in a tiny space. Some classic path planning algorithms such as RRT and A* can also be applied to autonomous parking. DO-RRT [5] combines a magnetic-like field model with original RRT in order to improve the planning efficiency in obstacle clustered environments. Heuristic searching, for instance, OSEHS [6] use oriented circle to explore free space in narrow places and generate orientation knowledge, which can accelerate the heuristic searching process significantly.

As for a parking process, a series of maneuvers would be carried out in a tiny region, therefore, a precise path planning and motion control are required. A simple and low computational cost path planning algorithm named geometric method shows effectiveness in autonomous parking. Premier work can be traced back to 1950s when Dubins [7] proposed a method to generate paths by using straight line segments and circle arcs with the minimum turning radius of vehicle. This method was improved by Reeds and Shepp [8], who took both forward and backward motion into consideration. Kim et al. [9] proposed a practical method which can generate paths in cluttered environments by connecting tangential circular arcs and a straight line, moreover, the parking path can be easily optimized by minimizing a cost function. Sungwoo et al. [10] applied the circular locus method to parallel parking, and proposed a robust path following strategy by using robust grey-box control. The biggest disadvantage of this method is the discontinuity of the curvature at the junction points. In order to track the path precisely, the vehicle has to stop at the junction points and reorient the front wheels. This will unnecessarily cost time and damage steering system. Many methods have been proposed so as to avoid these problems, among which the Euler spiral has a unique advantage in dealing with path smoothing. Fraichard et al. [11] used clothoid arc as a complement of Reeds and Shepp's locus to build up smooth paths, and Vorobieva et al. [12] applied the path planning method in [11] to autonomous parking system and verified its feasibility. However, this method has to firstly generate paths with circle arcs, then refine the result with a clothoid sequence, which needs extra computational time.

In order to enlarge the flexibility in various environments, we proposed a geometric method based on smooth path searching and optimizing. This method firstly evaluates the parking environment, then searches the path from the desired parking slot to the initial configuration reversely. A group of paths can be generated in both obstacle-free and cluttered scenarios, and the optimal path can be selected in the optimization process. Benefited from the geometry-based method, reference control signals can be deduced directly once the path is successfully generated. In path tracking procedure, we design a simple feedforward controller in order to deal with the uncertainty of the vehicle itself and the disturbance from the external environment.

The remainder of this paper is organized as follows. Section II focuses on the basic problems of parking path planning. In section III, we introduce the formation of

This work is partly supported by National Natural Science Foundation of China (Grant No. NSFC 61473042 and 61105092), Beijing Higher Education Young Elite Teacher Project (No.YETP1215) and Program for Changjiang Scholars and Innovative Research Team in University (IRT-16R06, T2014224). [1]School of Automation and State Key Laboratory of Complex System Intelligent Control and Decision, Beijing Institute of Technology, Beijing, China. [2]School of Automation, Tsinghua University, Beijing, China. *Yang Yi is the corresponding author (phone: 86-10-68913985; fax: 86-10-68918381; e-mail: yang_yi@bit.edu.cn).

the linearly steering spiral, which is generated from the kinematics of the car-like vehicle. Then, the path planning algorithm is detailed in section IV. Finally, experimental results and conclusions are given in section V and section VI respectively.

II. Preliminary

During the path planning process, it is necessary to take the model constraints into consideration. Due to the dynamic model is less relevant in low-speed scenarios such as parking, in this paper we consider a conventional kinematics of a car-like vehicle.

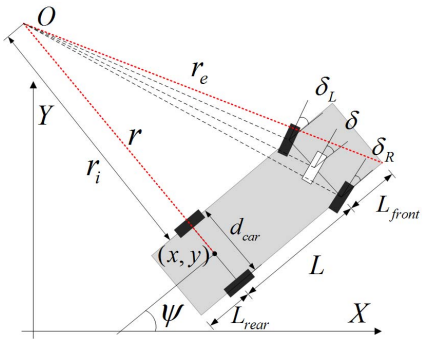


Fig. 1. Ackermann steering geometry

Fig. 1 shows the Ackermann steering geometry. We use the center of rear axle to describe the position of the vehicle in the world coordinate while ψ represents the angle between vehicle orientation and positive direction of X axis in counterclockwise. Other parameters of the vehicle are listed in TABLE I. According to the non-holonomic constraints, the kinematics of car-like vehicle can be expressed as

$$\begin{aligned} \dot{x} &= v \cdot \cos\psi \\ \dot{y} &= v \cdot \sin\psi \\ \dot{\psi} &= v \cdot \tan\delta / L \end{aligned} \tag{1}$$

wherein v denotes the instantaneous longitudinal velocity.

TABLE I
Notation of Vehicle Parameters

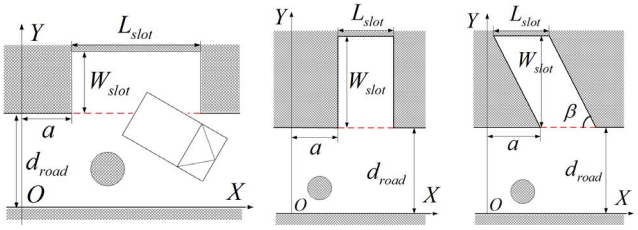
Parameter	Description
L_{car}	Length of the vehicle
d_{car}	Width of the vehicle
L	Length of wheelbase
L_f/L_r	Front/Rear overhang
δ_l/δ_r	Steering angle of the front left/right wheel
δ	Steering angle of the front virtual wheel
r	Turning radius with respect to center of rear axle
r_i	Turning radius with respect to rear inner wheel
r_e	Turning radius with respect to front outer vertex

Moreover, interior constraints such as mechanical and physical constraints of car-like vehicle should be taken into consideration, namely

$$|v| \in [0, v_{max}], |\delta| \in [0, \delta_{max}], |\dot{\delta}| \in [0, \dot{\delta}_{max}] \tag{2}$$

wherein v_{max} represents maximum velocity of the vehicle. δ_{max} stands for maximum steering angle which is restricted to the structure of steering mechanism. $\dot{\delta}_{max}$ denotes the maximum steering velocity which depends on the capability of EPS (Electric Power Steering) system.

Parking environment usually consists of two parts, namely structured parking area and irregular obstacles. Structured parking area always includes an available parking space and the road aside. The most common conditions are parallel parking, garage parking and angle parking as shown in Fig. 2. In addition, we use circles to represent the occupancy area of the obstacles in environments.



(a) Parallel parking (b) Garage parking (c) Angle parking

Fig. 2. Three typical parking environments, wherein L_{Slot}, W_{Slot} are length and width of the parking slot, d_{road} is the width of the road aside.

III. Linearly Steering Spiral

Circular locus method, i.e. geometric algorithms based on circle arcs and segmental lines, is widely used in autonomous parking. However, the discontinuity of the curvature on the junction points (we call it step points for convenience) brings difficulties to the path tracking procedure. Therefore, we hope to find a path model whose curvature is smooth, meanwhile, can describe the vehicle motion explicitly.

Assuming a car-like vehicle is driving along a straight line from A as shown in Fig. 3, then it start to turn left at B. During the turning process, the steering angle of the front wheel is proportional to the travel distance until reaching the desired angle at C. The transitional path $\overline{BC}$ connects the line and arc, thus there is no step point on this trajectory.

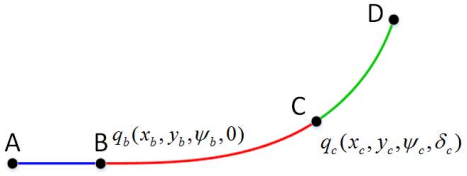


Fig. 3. A smooth path without step point.

As mentioned in the previous section, the motion of vehicle follows the kinematics (1). We can translate (1) into space-domain by using $ds = v \cdot dt$, and express it in an incremental form,

$$\begin{aligned} x(s) &= x_0 + \int_0^s \cos(\psi_0 + \Delta\psi(s))ds \\ y(s) &= y_0 + \int_0^s \sin(\psi_0 + \Delta\psi(s))ds \\ \psi(s) &= \psi_0 + \Delta\psi(s) \end{aligned} \tag{3}$$

wherein $\Delta\psi(s)$ denotes the orientation variation of the vehicle.

In order to couple the longitudinal and lateral control, let $\dot{\delta}/v$ be a constant,

$$\frac{d\delta}{dt}/\frac{ds}{dt} = \frac{d\delta}{ds} = \lambda \tag{4}$$

wherein, λ reflects the relationship between steering angle and travel distance. This relationship is uncorrelated to time variation, thus once λ is determined, the shape of the curve will not change.

Because the change of instantaneous radius r is continuous with respect to the travel distance, we can transform (4) into

$$\frac{d\delta}{ds} = \frac{d\delta}{dr}\cdot\frac{dr}{ds} = -\xi(\delta)\cdot\frac{L}{r^2+L^2}\cdot\frac{dr}{ds} \tag{5}$$

wherein, $\xi(\delta)$ denotes the sign of steering angle. Then,

$$\frac{dr}{ds} = -\xi(\delta)\cdot\left(\frac{\lambda}{L}\cdot r^2 + L\cdot\lambda\right) \tag{6}$$

Thus, we can obtain the representation of r by solving the differential equation above,

$$r = L\cdot\tan\left(\arctan\left(\frac{r_0}{L}\right) - \xi(\delta)\cdot s\lambda\right) \tag{7}$$

wherein, r_0 denotes the turning radius at configuration q_b. The variation of ψ from q_b can be deduced by using $\Delta\psi = \int\frac{1}{r}ds$,

$$\Delta\psi = -\frac{1}{L\lambda}\ln\left|\sin\left(\arctan(\frac{r_0}{L}) - \xi(\delta)\cdot s\lambda\right)\right| \tag{8}$$

At this point, the representation of the clothoid-like curve is generated. For convenience, we denote the proposed curve as linearly steering spiral (LSS). Hence, we can use these basic elements, namely, segmental line, circular arc, and LSS to generate a smooth path. However, the computational cost of LSS is relatively expensive, so in this paper, we calculate the LSS and store the results in a lookup table once λ is determined in order to improve the computational efficiency in the path planning procedure.

IV. Algorithm Description

A. General Overview

According to the kinematics of car-like vehicle, we can easily find that the shape of trajectories are only related to steering angle rather than the sign of longitudinal velocity. Therefore, in this paper, we generate paths by retrieving the vehicle from the parking slot.

The overall workflow of the proposed method is depicted in Fig. 4. Firstly, a series of preparations are executed, including setting up a coordinate system and parameters initialization. We evaluate the parking environment to detect whether the vehicle could enter the desired parking slot in one trial, and we use different methods to calculate the target zone according to the evaluation result. Then, for each target point, we try to retrieve the vehicle out of the parking slot in the preparation motion. After that, an iteration process is carried out. Template calculation verifies whether there is any path connecting the initial and current configurations. If there is no path, the current state will be extended by intermediate motion, until the template calculation successes or the iteration depth reaches the maximum value. The optimization process can select an optimal path from all the generated paths. Finally, we deduce the reference control signal sequences from the optimal path.

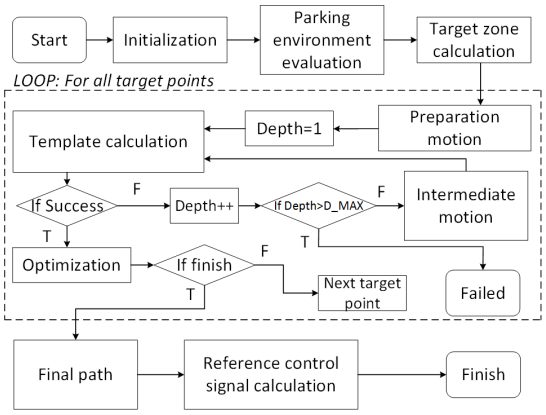


Fig. 4. Workflow of the proposed method.

The main idea of the proposed method is same in parallel parking, garage parking, and angle parking. Thus, without loss of generality, we introduce the path planning on parallel parking problem in this chapter.

B. Parking Slot Evaluation and Target Zone Calculation

As for parallel parking, whether the vehicle can finish parking in one trial or not depends on the vehicle kinematics and the length of the parking slots. Here, we classify the parking slots into three categories.

The first case is that the length of the parking slot is big enough, so the vehicle can exit the parking slot starting with a piece of LSS. As shown in Fig. 5(a), the steering angle of the vehicle is 0 at A, and reaches the maximum value δ_{max} at B. We can deduce the minimum length of the parking slot in this case,

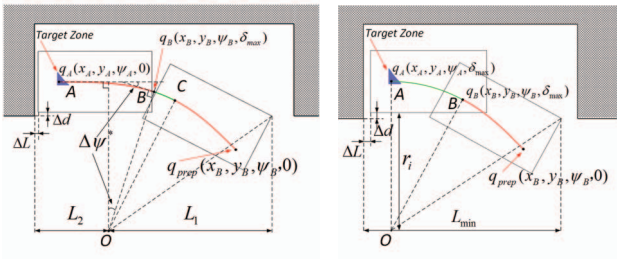
$$L^*_{min} = L_1 + L_2 \tag{9}$$

wherein,

$$\begin{aligned} L_1 &= \\ &\sqrt{r_e^2 - \left(r_i\cos\Delta\psi + \Delta y + \frac{d_{car}}{2}(\cos\Delta\psi - 1) - \Delta d\right)^2} \\ L_2 &= L_r + \Delta L + \Delta x + \frac{d_{car}}{2}\Delta\psi - r_i\sin\Delta\psi \end{aligned} \tag{10}$$

In the equations above, Δx, Δy, $\Delta\psi$ denote the variation of the state variables of the vehicle during LSS. ΔL denotes the distance between the rear edge of the vehicle and the back edge of the parking slot while Δd denotes the distance between the outer edge of the vehicle and the road edge.

The second case is that the length of the parking slot is smaller than L^*_{min} as shown in Fig. 5(b), but the vehicle can



(a) 1st case (b) 2nd case

Fig. 5. Two cases for parking in one trial.

exit the slot by driving with maximum steering angle in one trial. The minimum value of the parking slot length in this case is

$$L_{min} = \Delta L + L_r + \sqrt{(r_{min} + \frac{d_{car}}{2})^2 + (L_{car} - L_r)^2 - (r_{min} - \frac{d_{car}}{2} - \Delta d)^2} \tag{11}$$

The third case is that the length of the parking slot is smaller than L_{min}. In this case, the vehicle has to exit the parking slot with several trials. This process is well discussed in [12].

After the case is determined, we can calculate all the available target points in the desired parking slot. If the vehicle can exit the slot in one trial, the target zone can be calculated by using L^*_{min} and L_{min}. The blue parts in Fig. 5(a) and 5(b) represent the target zones. If the vehicle has to exit the parking slot in several trials, we choose the point next to the road edge and rear boundary as the target point in order to obtain the least execution time and fewest maneuvers.

C. The Path-finding Strategy

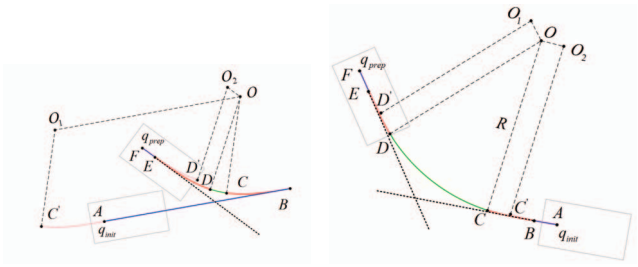
Before the path-finding procedure, an array of points will be sampled in the target zone. The path-finding procedure starts from every sampled points and generates a group of available paths to the initial configuration. The procedure mainly contains three steps, i.e. preparation motion, template calculation and intermediate motion. The main work of each step is as follows.

1) Preparation motion: Preparation motion is the first step of path-finding, and it will be only executed once for each target point. The goal of this step is to move the two front vertices of the vehicle out of the parking slot. According to previous analysis, preparation motion for different slot length is different. When $L > L^*_{min}$, the vehicle has enough operation space, a piece of LSS can be considered as the first motion in priority. For $L_{min} < L < L^*_{min}$, in order to exit the parking slot in one trial, so a circle arc can be considered in priority. As for $L < L_{min}$, the vehicle can not drive out of the slot in one trial, so it has to move forward and backward with maximum steering angle until the front vertices are out of parking slot. Then, an LSS path is generated so as to adjust the steering angle to zero. For convenience, we use q_{prep} to represent the vehicle configuration after the preparation motion.

2) Template calculation: After the preparation motion, we try to connect q_{prep} and the initial configuration q_{init} by a path template. Here, we use the sequence of basic elements: line-LSS-arc-LSS-line. Considering the layout of the parking environment is relatively simple and the width of the road is usually small, so we tend to generate path in a space-saving way. Fig. 6 shows two common cases of the path template. Since q_{init}, q_{prep} and λ are known, there are infinite solutions for the path template unless the radius of the circular arc $\overline{CD}$ is determined. Hence, the waypoints of the template can be calculated as follows.

Firstly, we set an empirical value for radius OC, then translate LSS $\overline{CB}$ and $\overline{DE}$ to point A and point F. Therefore, four parallelograms, $\overline{C'COO_1}$, $\overline{C'ABC}$, $\overline{D'DOO_2}$ and $\overline{FEDD'}$, are generated. C' and D' can be calculated from the LSS lookup table, and then, D', O_1, O_2 and waypoints B, C, D, E can be directly inferred according to the properties of parallelograms.

However, in actual situations, steering and shifting gear can be executed simultaneously in order to save time. Thus, it is acceptable that we can leave out LSS paths if the heading direction changes in the next motion if necessary. Moreover, in some extreme scenarios, it is more important to generate available paths rather than guarantee the smoothness. Thus, calculating a template of line-arc-line will replace the former one. Under this circumstance, the proposed method degrades into the conventional circular locus method.



(a) (b)

Fig. 6. Two typical formation of the path in template calculation.

3) Intermediate motion: If template calculation failed, the intermediate motion will be carried out. Intermediate motion randomly generates a collision free smooth path by using a piece of arc or a LSS-arc-LSS template so as to extend the searching area. This procedure will be executed iteratively until the template calculation successes or the iteration time reaches the upper limit. Herein, the value of iteration depth represents the extensibility of the path in some way. Big iteration depth will enlarge the distribution area of the paths, however, the more computational cost is needed. Generally, one or two iterations are enough for the proposed algorithm to find an available path in simple scenarios while three or four iterations for cluttered scenarios. Therefore, the maximum iteration depth could be determined after the environment evaluation.

D. Optimization

After the path-finding procedure, a group of available paths is generated. In order to select an optimal path, a cost function evaluating the smoothness, effectiveness, and safety of the generated paths is proposed. The cost function is designed as

$$C = \varepsilon D_{total} + \alpha / d_{obs} + \beta N_{steer} + \gamma N_{stop} + \eta N_{manu} \quad (12)$$

wherein, D_{total} represent the total length of the generated path and d_{obs} denotes the minimum distance between the vehicle and obstacle in the whole parking path. N_{steer}, N_{stop} and N_{manu} denote the times of steering, stop, maneuvers of the generated path, respectively. Therefore, the optimal path can be generated according to the cost function. As shown in Fig. 7(a) and Fig. 7(b), all available paths are shown in gray while the optimal paths are shown in color. Moreover, the proposed algorithm can also be applied in garage parking and angle parking, and the results are presented in Fig. 7(c) and Fig. 7(d).

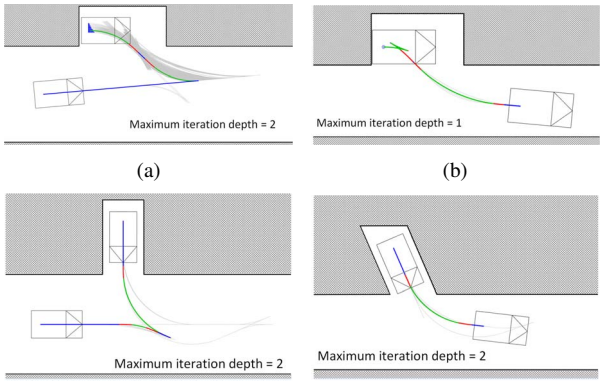


Fig. 7. Path planning results in different scenarios. (a), (b) is simple scenarios. The garage parking and angle parking are presented in (c) and (d) respectively.

E. The Path Tracking Controller

In order to deal with the uncertainty of the vehicle itself and the disturbance from the external environment, a simple feedforward closed-loop controller is designed.

We firstly generate the reference control sequences from the generated path. The longitudinal target velocity is designed as a trapezoid signal during a one-way movement. The target steering angle is correlated to the type of the ongoing path element, namely,

1) Moving along a straight line:

$$\delta(s) = 0, s \in [0, S_l] \quad (13)$$

wherein, S_l denotes the length of the segmental line.

2) Moving along a linearly steering spiral:

$$\delta(s) = \delta(0) + \xi(\delta) \cdot s \cdot \lambda, s \in [0, S_{lss}] \quad (14)$$

wherein, $\xi(\delta)$ denotes the sign of the steering angle, S_{lss} denotes the length of the LSS.

3) Moving along a circular arc:

$$\delta(s) = \arctan(L/R), s \in [0, S_{arc}] \quad (15)$$

wherein, R and S_{arc} denote the turning radius and the length of the on-going circular arc.

Then, we use the reference control signal as the feedforward term and add it into a PD controller described below,

$$\begin{aligned} a_{set} &= v_{ref} \cdot K_{ff} + e_s \cdot K_{p1} + \dot{e}_s \cdot K_{d1} \\ \delta_{set} &= \delta_{ref} + e_{dir} \cdot K_{p2} + \dot{e}_{dir} \cdot K_{d2} \end{aligned} \quad (16)$$

wherein, a_{set} and δ_{set} are the longitudinal and lateral control inputs, v_{ref} and δ_{ref} are the reference control signal, e_s and e_{dir} are the error of travel distance and orientation, K_{ff} is the feedforward coefficient of the longitudinal control and K_{p1}, K_{d1}, K_{p2}, K_{d2} are coefficients of the PD controller.

V. Experiments

In this section, the proposed algorithm is firstly analyzed in a virtual parking lot by using a robotics simulator and an online experiment on a real-world autonomous vehicle is presented.

A. Simulation Environment

We use V-REP (Virtual Robot Experimentation Platform) as the simulator to test the feasibility of the proposed method. The optimal path and control signals are calculated by control core written in C++, while all the physical interactions are simulated by V-REP. We use the remote API client to achieve communication between the controller and V-REP. The experimental data are collected by control core and analyzed offline in MATLAB. The simulation experiments are executed on an Intel Core i7-3632QM CPU with 8GB RAM running at 2.2 GHz.

In preparation for the real-world experiment, here we use a virtual model of IN^{2}BOT, a self-driving platform which is refitted from Polaris Ranger all-terrain vehicle. Parameters of IN^{2}BOT are shown in TABLE II.

TABLE II
GEOMETRIC PARAMETERS OF IN^{2}BOT

Parameter	Symbol	Value
Length	L_{car}	$308(cm)$
Width	d_{car}	$165(cm)$
Front Overhang	L_f	$60(cm)$
Rear Overhang	L_r	$55(cm)$
Maximum Steering Angle	δ_{max}	$26(^\circ)$

We build a virtual parking lot in the simulation environment (shown in Fig. 8) which contains 3 parallel parking slots with same width and different length. The parking slots are shown in different color while the road boundaries are represented in light green. According to previous chapters, two critical values L^*_{min} and L_{min} hinge on the maximum longitudinal and steering velocity. Once we set up v_{max} and $\dot{\delta}_{max}$, the values of λ, L^*_{min} and L_{min} can be inferred immediately. Related parameters are settled as shown in TABLE III. Under this circumstance, the lengths of the three

parking slots are designed as 450 cm, 550 cm, 650 cm, and the width of the slots are all 250 cm.

For path tracking, reference control signals are calculated and dispatched to the simple feedforward closed-loop controller introduced in previous section. And ground truth motion trajectories are acquired from the simulation software.

TABLE III
RELATED PARAMETERS FOR PARKING SIMULATIONS

v_{max}	$\dot{\delta}_{max}$	λ	L_{min}	L^*_{min}
$0.8m/s$	$10°/s$	$12.5°/m$	$504.7cm$	$608.9cm$

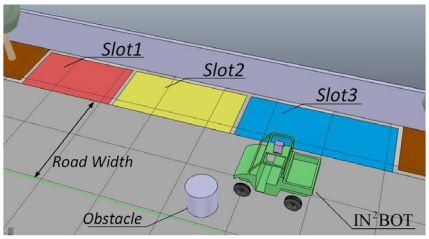


Fig. 8. Virtual parking lot in V-REP.

B. Simulation Results

Firstly, we test the proposed algorithm in several easy scenarios with no obstacle, and three groups of experimental results are shown in Fig. 9. Herein, the parking process is demonstrated in the upper left corner of each group, and the upper right figure shows the comparison between the generated reference path (red dashed curve) and ground truth trajectory (blue solid curve) recorded by the simulator. The coverage area during parking is shown in light blue. Vehicle status including variations of longitudinal velocity, steering angle and orientation are plotted. It can be seen that, the proposed method achieved good performance in these scenarios, continuous-curvature paths and smooth control signals are generated, and the vehicle can enter into the desired parking slot safely. With the increasing of the slot length, control signals are much smoother and the total time costs decrease. Moreover, the accuracy of the result is quite attractive, average distance and orientation error at the final poses of these tests are only 6.34 cm and 1.04 degrees respectively.

Parking simulation in cluttered environment is also tested and the results are shown in Fig. 10. Here, we put larger weights on smoothness in the optimization process, therefore, it tends to generate smooth path so that the length of the path is relatively longer. However, path tracking errors do not expand much as the distance grows owe to the closed-loop controller. The distance and orientation error at the final poses are 10.77 cm and 1.26° respectively.

In order to have a better comparison of the proposed method, we also applied a classical circular locus method within simulations, and the results are shown in Fig. 11. In this scenario, the paths generated by two methods are quite similar and both of them achieve good accuracy during the

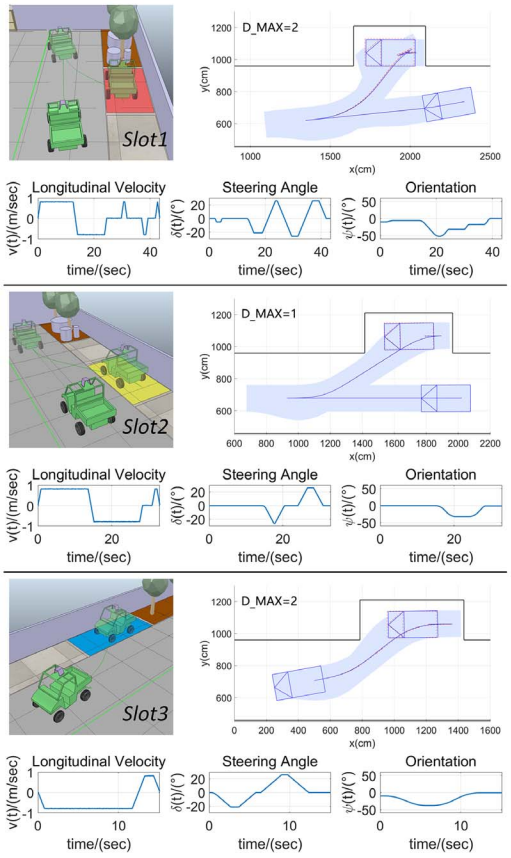


Fig. 9. Several results on easy scenarios, wherein D_MAX denotes the maximum iteration depth.

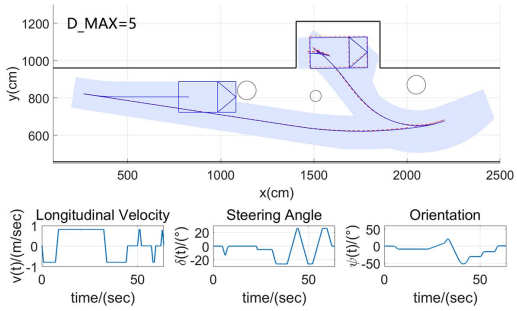


Fig. 10. Parking simulation in cluttered environment.

path tracking procedure. Nevertheless, time consumption of the proposed method is much smaller due to the difference of path smoothness. There are 4 step points on the path generated by circular locus method while only 1 step point for proposed method, which directly leads to a 33% time consumption improvement.

As for computational time consumption, smooth path planning in simple scenarios can be completed within 2.5 seconds by using C++ programming, while in the cluttered environment (scenarios in Fig. 10), this number reaches up to 10.4 seconds. Although there are more computational costs than circular locus method, it is tolerable for common usage in simple scenarios.

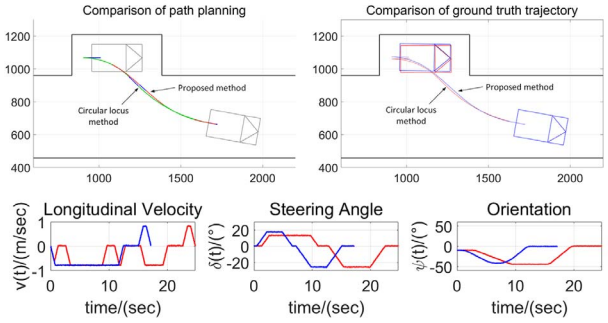


Fig. 11. Comparison between the circular locus method and the proposed method.

C. Real-world Experiments

The vehicle used for the real-world experiment is IN^{2}BOT, which is an autonomous driving prototype vehicle. All the calculations such as environment perception and motion planning are executed on two vehicle-mounted computers. The ground truth is acquired by an integrated navigation system with RTK and high accuracy IMU, whose precision reaches up to 2 centimeters.

The test is executed in a parallel parking scenario shown in Fig. 12, in which the length and width of the parking slot are 550 cm and 230 cm respectively. The experimental results are shown in Fig. 13.

Fig. 12. Real-world parking experiment.

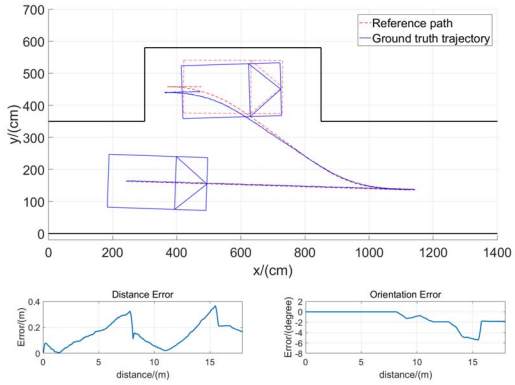


Fig. 13. Real-world parking experiment results.

The results shows that the vehicle can enter into the parking space successfully and safely. The final position and orientation errors are only 0.17 m and 1.84 degrees respectively, and they are well restrained within 0.4 m and 6 degrees during the entire parking process. Therefore, it can be seen that it is easy to track the parking path generated by the proposed method just by using a simple controller.

VI. Conclusions

In this paper, a path planning method for autonomous parking is proposed. We present a new basic path element, i.e. linearly steering spiral, which can describe the effect on the vehicle motion caused by the steering angle explicitly. Then, a path finding algorithm is proposed to generate a group of paths and an optimal path can be determined by using a designed cost function. Moreover, in order to deal with the uncertainty of the vehicle itself and the disturbance from the external environment, a simple feedforward closed-loop controller is designed. The proposed method is tested in the simulation environment, and a real-world online experiment based on an autonomous driving prototype vehicle is executed. The experimental results show good performance on both accuracy and computational cost.

In the future, we will integrate the proposed algorithm with heuristic searching methods so as to improve the efficiency of pathfinding strategy. In addition, research on online path regeneration will be carried out in order to improve the robustness.

References

[1] I. E. Paromtchik and C. Laugier, "Autonomous parallel parking of a nonholonomic vehicle," in *Proceedings of Conference on Intelligent Vehicles*, Sep 1996, pp. 13–18.

[2] D. Lyon, "Parallel parking with curvature and nonholonomic constraints," in *Proceedings of the Intelligent Vehicles '92 Symposium*, Jun 1992, pp. 341–346.

[3] T. H. S. Li and S.-J. Chang, "Autonomous fuzzy parking control of a car-like mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 33, no. 4, pp. 451–465, July 2003.

[4] G. Lini, A. Piazzi, and L. Consolini, "Multi-optimization of η^3-splines for autonomous parking," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 6367–6372.

[5] S. Shin, J. Ahn, and J. Park, "Desired orientation rrt (do-rrt) for autonomous vehicle in narrow cluttered spaces," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4736–4741.

[6] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1148–1153.

[7] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[8] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards." *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.

[9] D. Kim, W. Chung, and S. Park, "Practical motion planning for car-parking control in narrow environment," *IET Control Theory Applications*, vol. 4, no. 1, pp. 129–139, Jan 2010.

[10] C. Sungwoo, C. Boussard, and B. d'Andréa Novel, "Easy path planning and robust control for automatic parallel parking," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 656–661, 2011.

[11] T. Fraichard and A. Scheuer, "From reeds and shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.

[12] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking in tiny spots: Path planning and control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 396–410, Feb 2015.