A Novel Scenario-based Path Planning Method for Narrow Parking Space

Kaixiong Li
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
imagine@sjtu.edu.cn

Jun-Guo Lu*
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
*Corresponding Author:
jglu@sjtu.edu.cn

Qing-Hao Zhang
Department of Automation
Shanghai Jiao Tong University
Shanghai, China
zhangqinghao@sjtu.edu.cn

2023 35th Chinese Control and Decision Conference (CCDC) | 979-8-3503-3472-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCDC58219.2023.10327190

***Abstract*—Automatic parking is an important field in autonomous driving research, and path planning is a crucial step in its realization. However, traditional path planning algorithms tend to fail when confronted with narrow parking space. In this paper, a novel scenario-based path planning Method is proposed for narrow parking space. First, the scenario decision algorithm is used to determine the parking scenario category: parallel parking or perpendicular parking. Then suitable intermediate state in configuration space is selected through the backtracking method and the local path of intermediate-goal is generated. After that, the initial-intermediate path planning is performed. Finally, the global path is obtained by concatenating the two local paths. Experiments are conducted to evaluate the performance of the proposed method and the results show that it can not only quickly generate a path that satisfies vehicle kinematic constraints in narrow parking space.**

***Index Terms*—autonomous driving, narrow parking space, path planning, hybrid A*.**

I. Introduction

With the development of industrialization and information technology, the global car ownership has reached 1.446 billion, of which China accounts for more than 22%, ranking first in the world. But with this comes the problem of limited parking resources and frequent traffic accidents. On the other hand, the trend of car intelligence is unstoppable, and automatic driving is the key to car intelligence. As one of the most important scenarios of autonomous driving, automatic parking has been attracting the attention of academia and industry. If the automatic parking algorithm can safely and efficiently complete the task, even in narrow parking space, it would alleviate the current parking problem to some extent. Besides, sufficiently safe algorithms can also reduce the number of accidents and create economic value for society.

The autonomous driving software system is divided into four main parts: perception [1], localization [2], planning [3] and control [4]. In this paper, we focus on the path planning module, which generates a path from the initial state to the parking goal state, even if the parking space is narrow. Common path planning algorithms are usually divided into the following four categories: sampling-based planners [5]–[7], search-based planners [8]–[10], interpolating curve planners [11]–[14] and numerical optimization planners [15]. Sampling algorithms are characterized by uniform random sampling of the safe region in configuration space, represented by probabilistic poadmap method (PRM) [5] and rapidly-exploring random tree (RRT) [6]. Search-based methods' basic idea is to discretize the configuration space into a graph in a certain way, and then search for feasible solutions through various search algorithms, such as Dijkstra [8] and A* [10]. The interpolation fitting algorithms generate the feasible path using data interpolation and curve fitting, which can guarantee smoothness [13] or satisfy the non-holonomic constraint of vehicle [12]. Lastly, optimization methods find optimal trajectories by establishing objective functions and constraint relationships. For example, EM planner [15] is to model the path planning as a quadratic problem by taking the obstacle constraints into the Frenet coordinate system. In addition, some fusion methods also achieve good results in path planning. For example, RRT* with RS [16] algorithm has the characteristics of both fast RRT [6] and smooth RS curve [12]. Moreover, many algorithms have been studied and proposed for parking scenarios [17]–[19].

Many researchs on path planning for narrow parking Spaces have been conducted. An improved A* algorithm is proposed to generate feasible path of narrow perpendicular parking space [20]. The anchor point close to parking goal point is obtained by fast A* algorithm. Then "Go and Turn" strategy is utilized to park the vehicle from the anchor point into goal pose. However, using RS curves [12] to connect initial state and anchor point is not applicable to large-scale complex scenarios. In addition, a path planning method based on clothoid curve is presented to reduce unnecessary maneuvers for narrow perpendicular parking space [21]. But this method only works when the vehicle is already close to the parking space. For narrow parallel parking space, an optimization-based trajectory planning algorithm is introduced to complete path planning and speed planning [22]. Nevertheless, when the environment is complex, such as when the obstacles near narrow parking space are irregularly placed and difficult to fit accurately in terms of shape and size, the modeling process becomes complicated and may even become unsolvable. **To the best of our knowledge, there is currently no path planning method that can uniformly handle narrow paral-**

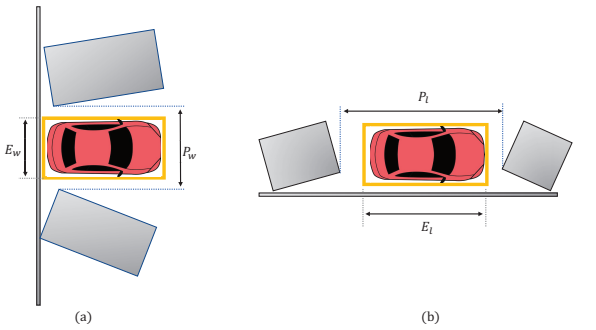


Fig. 1. SCS for different parking scenarios. (a) Perpendicular parking. (b) Parallel parking.

lel and perpendicular parking space while relying less on environmental modeling.

Motivated by the above mentioned discussions and facts, a novel scenario-based path planning method that can be applied to both narrow parallel and perpendicular parking scenarios is proposed in this paper. The obstacles and the environment information are only represented by the occupancy grid map, so there is no need to consider the geometric features of the obstacles for modeling. The scenario decision algorithm is firstly presented. And then appropriate intermediate state is selected by backtracking method which generates local path from intermediate state to goal state. Then former path is calculated to connect the initial state and the intermediate state. The final path is formed by stitching together two local paths.

II. Problem Formulation

Common parking scenarios today include on-street parallel parking and perpendicular parking in parking lots. Although there are generally parking space markings to indicate the spot, the parking space can become narrow when the surrounding vehicles are parked irregularly. Here we use smallest constraint space(SCS) [23] as narrowness measure of parking space, which are defined as:

$$\begin{aligned} SCS_{para} &= P_l/E_l, \\ SCS_{perp} &= P_w/E_w, \end{aligned} \tag{1}$$

where E_l, E_w represent the length and width of the vehicle respectively. P_l is the length of free space for parallel parking spot, and P_w is the width for perpendicular parking spot. Scenarios are shown as Figure 1.

We adopt the traditional bicycle model to describe the four-wheel vehicle. And $\boldsymbol{e} = (x_e, y_e, \theta_e)$ represents the vehicle pose, where (x_e, y_e) is the position of the rear axle center and θ_e denotes the angle of direction. Other parameters of vehicle include wheel base E_{wb}, maximum steering angle ϕ_{max}. According to the model properties, the maximum curvature of the vehicle's path can be expressed as $tan(\phi_{max})/E_{wb}$

The focus of this article is on the problem of path planning for a vehicle to reach a narrow parking spot, considering both parallel and perpendicular parking scenarios. Therefore, the required information only includes the initial state

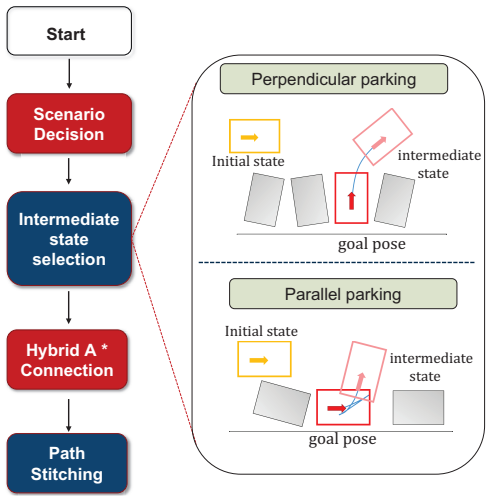


Fig. 2. Scenario decision based path planning framework.

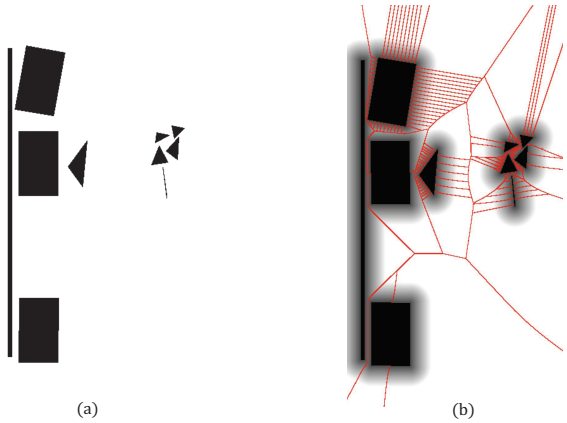


Fig. 3. Parallel parking environment. (a) Occupancy grid map. (b) Euclidean distance map generated from occupancy grid map.

$\boldsymbol{i} = (x_i, y_i, \theta_i)$, the parking goal state $\boldsymbol{g} = (x_g, y_g, \theta_g)$, the occupancy grid map representing the environment and the basic parameters of vehicle.

III. Scenario-based Path Planning Method

To address the narrow parking issue outlined in Section II, the novel scenario-based path planning algorithm is proposed. As depicted in Figure 2, the algorithm consists of four parts. Scenario decision determines the type of parking scenario, and the intermediate state selection module selects an appropriate intermediate state based on the scenario and local path from the intermediate state to the goal state is obtained. Then Hybrid A* search is used to find a path from the initial state to the intermediate state. The final global path is obtained by stitching these two paths together.

A. Scenario Decision

As mentioned in Section II, the parking scenarios are divided into parallel parking and perpendicular parking. Goal

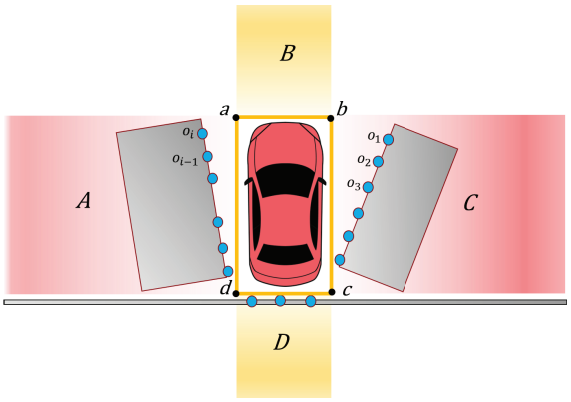


Fig. 4. Space division around parking spot. Blue points $o_1, o_2, \cdots, o_i$ are the nearest obstacle points.

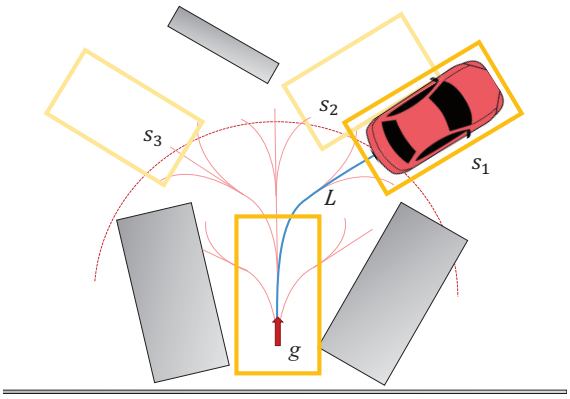


Fig. 5. Intermediate state selection method for perpendicular parking.

state $\boldsymbol{g}$ and occupancy grid map are provided without extra information about parking space, so it is necessary to determine the parking scenario. Euclidean distance map (EDM) [24] stores in each cell the distance to its closest obstacle, which can be generated from occupancy grid map. With the help of EDM, we can obtain the coordinates and distance of the nearest obstacle point of each point in the map, which is very important for scenario decision. In Figure 3, the occupancy grid map is on the left and the EDM is on the right. The red lines in EDM are Voronoi lines, which denote the collection of isometric points to the obstacle.

According to the goal pose, the space around the parking spot is divided into four parts: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$, representing the left, front, right and rear. As shown in Figure 4, the vertex coordinates of the rectangle box representing the vehicle are denoted as $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d} \in \mathbb{R}^2$. Here, the four space is formulated as half space intersection:

$$\begin{aligned}
\boldsymbol{A} &= \left\{\boldsymbol{p} \mid \boldsymbol{v_1}^{\mathrm{T}}\boldsymbol{p} > \sigma_4, \boldsymbol{v_2}^{\mathrm{T}}\boldsymbol{p} < \sigma_1, \boldsymbol{v_3}^{\mathrm{T}}\boldsymbol{p} > \sigma_2\right\}, \\
\boldsymbol{B} &= \left\{\boldsymbol{p} \mid \boldsymbol{v_2}^{\mathrm{T}}\boldsymbol{p} > \sigma_1, \boldsymbol{v_3}^{\mathrm{T}}\boldsymbol{p} < \sigma_2, \boldsymbol{v_4}^{\mathrm{T}}\boldsymbol{p} > \sigma_3\right\}, \\
\boldsymbol{C} &= \left\{\boldsymbol{p} \mid \boldsymbol{v_1}^{\mathrm{T}}\boldsymbol{p} > \sigma_4, \boldsymbol{v_3}^{\mathrm{T}}\boldsymbol{p} > \sigma_2, \boldsymbol{v_4}^{\mathrm{T}}\boldsymbol{p} < \sigma_3\right\}, \\
\boldsymbol{D} &= \left\{\boldsymbol{p} \mid \boldsymbol{v_1}^{\mathrm{T}}\boldsymbol{p} < \sigma_4, \boldsymbol{v_2}^{\mathrm{T}}\boldsymbol{p} > \sigma_1, \boldsymbol{v_4}^{\mathrm{T}}\boldsymbol{p} > \sigma_3\right\},
\end{aligned} \tag{2}$$

where

$$\begin{aligned}
&\boldsymbol{v_1} = \boldsymbol{a} - \boldsymbol{d}, \boldsymbol{v_2} = \boldsymbol{b} - \boldsymbol{a}, \boldsymbol{v_3} = \boldsymbol{c} - \boldsymbol{b}, \boldsymbol{v_4} = \boldsymbol{d} - \boldsymbol{c}, \\
&\sigma_1 = \boldsymbol{v_1}^{\mathrm{T}}\boldsymbol{d}, \sigma_2 = \boldsymbol{v_3}^{\mathrm{T}}\boldsymbol{b}, \sigma_3 = \boldsymbol{v_4}^{\mathrm{T}}\boldsymbol{c}, \sigma_4 = \boldsymbol{v_1}^{\mathrm{T}}\boldsymbol{d},
\end{aligned} \tag{3}$$

$\boldsymbol{p} \in \mathbb{R}^2$ represents any point in the space.

We sample the points on the rectangle box representing the vehicle, and then count the spatial distribution of the nearest obstacle points. Assuming that obstacle points are distributed in space $\boldsymbol{A}, \boldsymbol{C}$ and $\boldsymbol{D}$, the current scenario is perpendicular parking. The set of nearest obstacle points is denoted by $\boldsymbol{O} = \{\boldsymbol{o_1}, \boldsymbol{o_2}, \cdots, \boldsymbol{o_i}\}$. In this way, scenario decision conditions can be written as:

$$\begin{aligned}
&\text{Perpendicular parking}: \boldsymbol{O} \cap \boldsymbol{B} = \emptyset \quad or \quad \boldsymbol{O} \cap \boldsymbol{D} = \emptyset, \\
&\text{Parallel parking}: \boldsymbol{O} \cap \boldsymbol{A} = \emptyset \quad or \quad \boldsymbol{O} \cap \boldsymbol{C} = \emptyset.
\end{aligned}$$

Note that this is scenario decision for narrow parking space, so it will not be the case that all intersections are empty.

B. Intermediate State Selection

Due to the narrow parking space, the traditional path planning methods usually spend a lot of time near the goal point, and the planning even fails in the end. We use the backtracking method for reverse planning, starting from the parking goal spot, to plan a partial path that can reach the appropriate state, which needs to get rid of narrow space, namely the intermediate state mentioned in this paper.

1) Perpendicular Parking: For the narrow perpendicular parking, vehicle starts to move from the goal state and only needs to adjust the moving direction to reach multiple states without switching gears. As shown in Figure 5, we use the node extension idea of hybrid A* algorithm [9] in the perpendicular parking space, and the path forks after the vehicle advancing for a certain distance. When these extended paths grow to a limit distance $dist_{perp}$ from the parking spot, we can obtain the collision-free candidate state set $\boldsymbol{S} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \boldsymbol{s_3}, \cdots, \boldsymbol{s_i}\}$ after collision detection, where $\boldsymbol{s_i} = (x_{s_i}, y_{s_i}, \theta_{s_i})$.

These candidate intermediate states need to be evaluated to get as free of obstacles as possible. Again, EDM is utilized to help select the best intermediate state. Assuming vehicle reaches the candidate state $\boldsymbol{s_i}$, we sample the vehicle rectangle box. For each sampling point $\boldsymbol{q_{ij}}$, the distance $\varphi(\boldsymbol{q_{ij}})$ from the nearest obstacle in the map can be queried. Therefore, the distance between the vehicle and the obstacle is defined as:

$$\psi\left(\boldsymbol{s_i}\right) = \frac{\sum_{j=1}^{n} \varphi\left(\boldsymbol{q_{ij}}\right)}{n}. \tag{4}$$

Thus, the optimal intermediate state $\boldsymbol{s_{perp}}$ is the one that is furthest away from the obstacle:

$$\boldsymbol{s_{perp}} = \left\{\boldsymbol{s} \mid \psi(\boldsymbol{s}) \geq \psi\left(\boldsymbol{s_i}\right)\right\}, \quad \boldsymbol{s}, \boldsymbol{s_i} \in \boldsymbol{S}. \tag{5}$$

After $\boldsymbol{s_{perp}}$ is selected, the corresponding path from the parking spot $\boldsymbol{g}$ to $\boldsymbol{s_{perp}}$ can also be obtained, such as $\boldsymbol{L}$ in Figure 5.

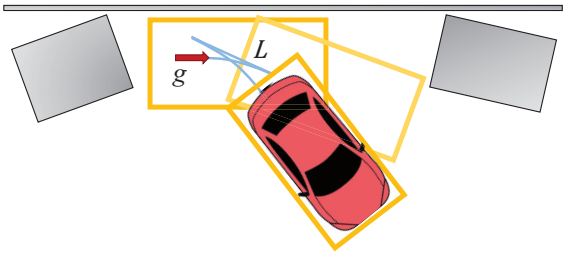


Fig. 6. Intermediate state selection method for parallel parking.

2) Parallel Parking: The strategy described above cannot be used in the narrow parallel parking space, because the vehicle needs to move forward and back many times to leave the parallel parking space. Therefore, we adopt the strategy of turning forward and straight back to drive the vehicle away from the goal pose.

In scenario decision part, the algorithm can not only distinguish the parking scenario, but also detect the relative position between the free space and the vehicle. As shown in Figure 6, the free space is located on the right side of the vehicle. We use "forward-backward" as a round of motion. When moving forward, the vehicle steers right with maximum angle ϕ_{max}, and the movement track is an arc. When encountering obstacles, the vehicle moves backwards in a straight line, i.e., the steering angle is 0. The "forward-backward" motion is repeated until the vehicle moves forward a specified distance $dist_{para}$ in each round without encountering any obstacles, and the vehicle is considered to have moved out of the narrow space, and the present pose is the optimal intermediate state $\boldsymbol{s_{para}}$. Similarly, when the free space is on the left side of the vehicle, the same strategy is used, except that the vehicle needs to turn left as it moves forward.

In addition to the intermediate state $\boldsymbol{s_{para}}$, the path from the goal state $\boldsymbol{g}$ to the intermediate state can also be obtained, as shown by $\boldsymbol{L}$ in Figure 6.

C. Hybrid A* connection

After the above steps, we have the intermediate state which is less constrained by obstacles, and the local path from the intermediate state to the goal state $\boldsymbol{g}$. As long as the feasible path from the initial state $\boldsymbol{i}$ to the intermediate state is obtained, the complete path can be generated by stitching of the two local paths.

We use hybrid A* algorithm [9] to search the other local path. Since the whole configuration space has been discretized in the selection of the intermediate state of perpendicular parking, the hybrid A* algorithm can make use of these steps that have been carried out to optimize the computing process. In fact, because of less constraint of intermediate state, any conventional planning algorithm can be used in this part, but in order to ensure the quality of the path, we choose hybrid A* algorithm. Here are the key steps of the hybrid A* algorithm we used:

1) Cost Function: Assume that the current extended node state is $\boldsymbol{e_i}$ and its parent is $\boldsymbol{e_{i-1}}$. Then the cost function of $\boldsymbol{e_i}$ is:
$$f\left(\boldsymbol{e_i}\right) = g\left(\boldsymbol{e_i}\right) + h\left(\boldsymbol{e_i}\right), \tag{6}$$
$g\left(\boldsymbol{e_i}\right)$ represents the cost from initial state $\boldsymbol{i}$ to $\boldsymbol{e_i}$, and $h\left(\boldsymbol{e_i}\right)$ is cost from $\boldsymbol{e_i}$ to intermediate state $\boldsymbol{s}$. More detailed definition about them are as follows:
$$g\left(\boldsymbol{e_i}\right) = \begin{cases} g\left(\boldsymbol{e_{i-1}}\right) + w_{\mathrm{F}} \cdot l\left(\boldsymbol{e_{i-1}}, \boldsymbol{e_i}\right), & \boldsymbol{e_{i-1}} \xrightarrow{\text{forward}} \boldsymbol{e_i}, \\ g\left(\boldsymbol{e_{i-1}}\right) + w_{\mathrm{B}} \cdot l\left(\boldsymbol{e_{i-1}}, \boldsymbol{e_i}\right), & \boldsymbol{e_{i-1}} \xrightarrow{\text{back}} \boldsymbol{e_i}. \end{cases}$$
$$h\left(\boldsymbol{e_i}\right) = \begin{cases} l_{RS}\left(\boldsymbol{e_i}, \boldsymbol{s}\right), & \left\|\boldsymbol{e_i} - \boldsymbol{s}\right\|_2 < b_h, \\ \left\|\boldsymbol{e_i} - \boldsymbol{s}\right\|_2, & \text{else}, \end{cases} \tag{7}$$
where $l\left(\boldsymbol{e_{i-1}}, \boldsymbol{e_i}\right)$ denotes the distance from the parent node $\boldsymbol{e_{i-1}}$ to $\boldsymbol{e_i}$. w_{F} and w_{B} represent the weights when the current node is expanded forward and backward by $\boldsymbol{e_{i-1}}$ respectively. Notice that $g\left(\boldsymbol{i}\right) = 0$. In addition, $l_{RS}\left(\boldsymbol{e_i}, \boldsymbol{s}\right)$ is the Reed-Shepp curve distance [12] from $\boldsymbol{e_i}$ to $\boldsymbol{s}$, and b_h is the constrain distance to intermediate state $\boldsymbol{s}$.

2) Analytic Expansions: When the extended node approaches the intermediate state $\boldsymbol{s}$, we also use the Reed-Shepp model [12] to generate the path and check collision. The search ends when a collision-free path is found. Here we set the distance threshold at the beginning of the analytic expansions to $b_h/2$.

In this way, local path $\boldsymbol{F}$ from the initial state to intermediate state can be quickly obtained through hybrid A*. Then the global path can be obtained by stitching two local paths, i.e., $\boldsymbol{F} + \boldsymbol{L}$.

IV. Experiment Results and Analysis

Two sets of experiments are designed to verify the performance of the proposed method. The first group is comparison experiment with other algorithms to show the improvement of our method in the computing time and the quality of the generated path. The second group is the ultimate performance test experiment to explore the narrowest parking scenario that proposed method can handle. All experiments were carried out on a platform with i7-12700KF CPU and 32G RAM. All algorithms were completed by C++14 without parallel acceleration, and developed based on ROS-Melodic.

A. Comparison Experiments

The proposed method is compared with two path planning method: traditional hybrid A* [9] and RRT* with RS [16]. Both of these methods can generate the feasible path that meets the requirements of vehicle nonholonomic constraint and obstacle avoidance. To be fair, the parameters in the hybrid A* are consistent with the hybrid A* connection part of our algorithm. We evaluate the computing time (**CT**) and the path length (**PL**) to show the algorithm performance. For RRT* with RS method, considering its random sampling characteristic, we take 100 sampling iteration as the basic unit of time record. The iteration times (**ITs**) is recorded as a

metric. Parameters used in the experiment are shown in Table I. The experiment scenarios are shown in Figure 7, with the size of $15m \times 15m$ both.

TABLE I
EXPERIMENT PARAMETERS

Parameter	Description	Setting
E_l	Vehicle length	2.55 m
E_w	Vehicle width	1.55 m
E_{wb}	Vehicle wheel base length	1.9 m
ϕ_{max}	Vehicle maximum steering angle	0.47 rad
SCS_{perp}	SCS of perpendicular parking	1.4
SCS_{para}	SCS of parallel parking	1.6
$dist_{perp}$	extension distance for perpendicular parking	$1.5 \times E_l$
$dist_{para}$	maximum distance in each round of parallel parking	1.0 m
w_{F}	weight value for forward expansions	1
w_{B}	weight value for backward expansions	2
b_h	constrain distance to intermediate state $\boldsymbol{s}$	8 m

TABLE II
COMPARISON OF DIFFERENT ALGORITHMS

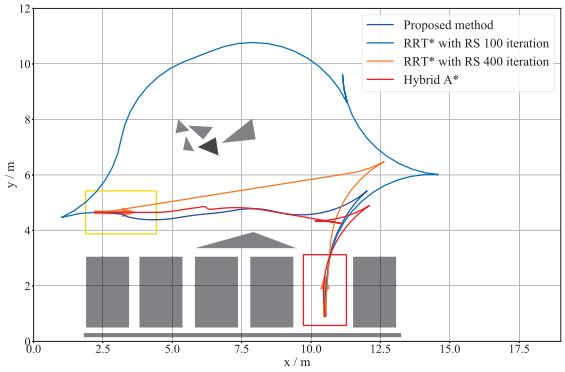
Scenario	Method	CT (ms)	PL (m)	ITs
Para-Parking	Proposed Method	**196.23**	**16.33**	—
	Hybrid A* [9]	18006.96	17.00	—
	RRT* with RS [16]	9783.27	18.36	500
Perp-Parking	Proposed Method	**217.83**	**15.03**	—
	Hybrid A* [9]	6127.27	16.76	—
	RRT* with RS [16]	710.06	29.54	100
		4206.53	16.91	500

Owing to scenario decision process, the proposed method correctly determined the experiment scenario and successfully planed the path. The experiment results are shown in Table II. It can be seen that the computing time of our method is greatly improved compared with other methods, and the path is also relatively superior. The algorithm is based on the kinematics characteristics of four-wheel vehicle, so the path can satisfy the nonholonomic constraint, as shown in Figure 7.

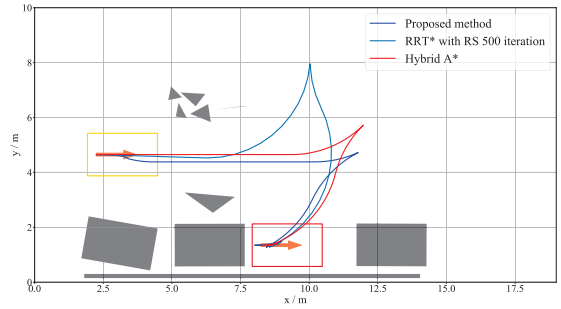
In fact, for parking task, although RRT* series algorithm is asymptotically optimal, due to the narrow parking space, it needs to be sampled several times to obtain a feasible solution, which is more difficult to optimize. In this parallel parking experiment, RRT* with RS finds a feasible solution after 500 iterations, but was still unable to optimize after 5000 iterations. The path planned by the hybrid A* algorithm is of better quality, but takes the longest time because too much time is spent on the end processing. Our method is aimed at this problem, using the idea of segmented processing, and finally achieved a good result.

B. Ultimate Performance Test Experiment

In the above comparison experiment, the narrowness measure of scenarios are SCS_{perp} and SCS_{para} respectively, but when SCS continues to become smaller, the traditional planning algorithm fail to work. We explored the proposed method's ultimate performance by constantly shrinking the parking space and recording the computing time. The experiment results are shown in Figure 8.

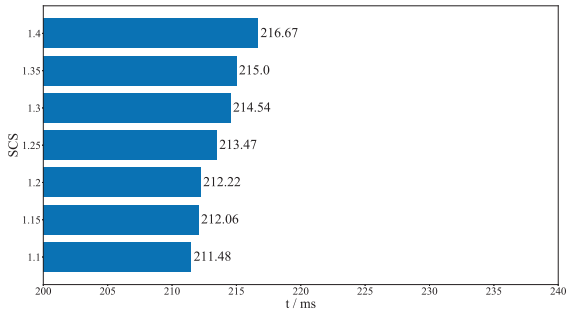


(a) Perpendicular parking.

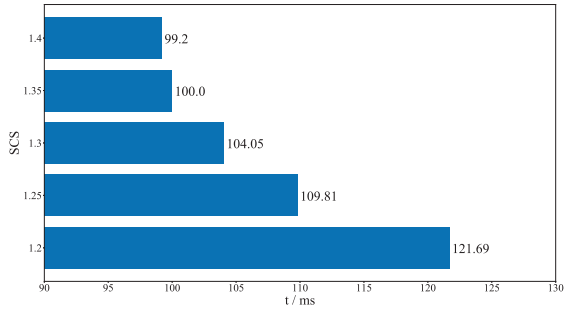


(b) Parallel parking.

Fig. 7. Experiment scenarios and path planning results.



(a) Perpendicular parking.



(b) Parallel parking.

Fig. 8. Ultimate performance test results of two scenarios.

For the perpendicular parking, it can be seen from Figure 8(a) that the computing time decreases with the reduction of SCS, but the change is not significant. On the one hand, when the parking space gets narrower, less candidate intermediate states are generated and the computing time become shorter. On the other hand, there is no need to switch gears when the intermediate state is generated. The car can always quickly drive away from the parking space, so the change of computing time is not very obvious. Therefore, as long as $SCS > 1$, perpendicular parking is always successful in our test environment.

Different from perpendicular parking, parallel parking requires multiple gear shifts to drive out of the parking space, and is constrained by ϕ_{max} during this process. As can be seen from Figure 8(b), when SCS gradually becomes smaller, i.e., the parallel parking space gets narrower, algorithm take longer to compute and vary more. When $SCS = 1.1$, path planning fails. After many tests, the limit of parallel parking performance is $SCS = 1.17$.

V. Conclusion

A scenario-based path planning algorithm for narrow parking space has been presented in this paper. In order to generate the global path, the backtracking method has been used to start from the goal state, employing different methods to select an appropriate intermediate state based on various scenarios. The local trajectory from the intermediate state to the goal state has been generated in this process. The Hybrid A* algorithm has been utilized to search for a path from the initial state to the intermediate state, and the global path has been composed of these two local paths. The proposed algorithm has been demonstrated to be able to quickly plan a qualified path in relatively abundant parking spaces through a comparison experiment, and it has been shown to be capable of handling the challenges of very narrow parking spaces through an ultimate performance test.

References

[1] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[2] Z. Huang and J.-G. Lu, "A new laser-based loop detection method for laneway environment 3d mapping," in *2021 China Automation Congress (CAC)*. IEEE, 2021, pp. 3219–3223.

[3] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.

[4] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[6] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[7] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

[8] J. Y. Hwang, J. S. Kim, S. S. Lim, and K. H. Park, "A fast path planning by path graph optimization," *IEEE Transactions on systems, man, and cybernetics-part a: systems and humans*, vol. 33, no. 1, pp. 121–129, 2003.

[9] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[10] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[11] 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.

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

[13] H. Vorobieva, N. Minoiu-Enache, S. Glaser, and S. Mammar, "Geometric continuous-curvature path planning for automatic parallel parking," in *2013 10th IEEE international conference on networking, sensing and control (ICNSC)*. IEEE, 2013, pp. 418–423.

[14] A. Simon and J. C. Becker, "Vehicle guidance for an autonomous vehicle," in *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No. 99TH8383)*. IEEE, 1999, pp. 429–434.

[15] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[16] Y. Dong, Y. Zhong, and J. Hong, "Knowledge-biased sampling-based path planning for automated vehicles parking," *IEEE Access*, vol. 8, pp. 156 818–156 827, 2020.

[17] J. Huang, Y. Yang, D. Ding, Y. Li, and Y. He, "Automatic parking paths planning research based on scattering points six-degree polynomial and easement curve," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, p. 09544070221076594, 2022.

[18] M. Kim, J. Ahn, and J. Park, "Continuous-curvature target tree algorithm for path planning in complex parking environments," *arXiv preprint arXiv:2201.03163*, 2022.

[19] Y. Liu and P. Wang, "An autonomous parking algorithm based on a-star algorithm correction and mpc path tracking," in *International Conference on Signal Processing and Communication Technology (SPCT 2021)*, vol. 12178. SPIE, 2022, pp. 544–549.

[20] J. He and H. Li, "Fast a* anchor point based path planning for narrow space parking," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 1604–1609.

[21] S. Sedighi, D.-V. Nguyen, and K.-D. Kuhnert, "A new method of clothoid-based path planning algorithm for narrow perpendicular parking spaces," in *Proceedings of the 5th International Conference on Mechatronics and Robotics Engineering*, 2019, pp. 50–55.

[22] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3263–3274, 2016.

[23] Z. Lin, Q. Li, Y. Liang, and D. Cheng, "Parallel parking algorithm based on autonomous path planning," *Application research of computers*, vol. 29, 2012.

[24] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of euclidean distance maps and voronoi diagrams," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 281–286.