Trajectory Planning for Autonomous Valet Parking in Narrow Environments With Enhanced Hybrid A* Search and Nonlinear Optimization

Jing Lian, *Member, IEEE*, Weiwei Ren, Dongfang Yang, Linhui Li, *Member, IEEE*, and Fengning Yu

***Abstract*—This article focuses on the problem of autonomous valet parking trajectory planning in complex environments. The task normally can be well described by an optimal control problem (OCP) for the rapid, accurate, and optimal trajectory generation. Appropriate initial guesses obtained by sampling or searching-based methods are important for the numerical optimization procedure. Still, in highly complex environments with narrow passages, it may incur high computational costs or fail to find the proper initial guess. To address this challenge, an enhanced hybrid A* (EHA) algorithm is proposed to address the issue. The EHA includes four steps. The first step is to quickly obtain the global coarse trajectory using a traditional A* search. The second step is constructing a series of driving corridors along the rough trajectory, then evaluating and extracting nodes from each wide or narrow passage based on the length of the box's side. The third step is to extract each passage's boundary points. The final step is connecting boundary points by hybrid A* and generating a feasible initial guess for OCP. To reduce safety risks, vehicles in particular areas (Fig. 1(b)) should travel as slowly as possible. The global and local speed restrictions are distinct, and local restrictions are only activated when the vehicle enters a particular area. This "if-else" structure makes the optimization problem difficult. A novel approximation formulation for these local state constraints is introduced to overcome this issue. The experimental results demonstrate that the proposed method for trajectory planning is effective and robust.**

***Index Terms*—Autonomous parking, trajectory planning, numerical optimization, collision avoidance.**

I. Introduction

A. Background

AUTONOMOUS driving techniques are crucial to developing a safe, environmentally friendly, comfortable, and convenient intelligent travel service system [1]. Typically, the primary modules of autonomous driving consist of perception, localization, prediction, planning, and control. Trajectory planning is one of the core modules whose purpose is to generate a safe and comfortable trajectory for autonomous vehicles to complete the driving task [2], where safety refers to avoiding collisions and comfort refers to avoiding abrupt changes in speed or acceleration. Proper trajectory planning plays a crucial role in enhancing the intelligence level of vehicles [3]. Current research on trajectory planning focuses primarily on structured highways [4], [5] and parking in unstructured scenes [6], [7], [8]. The primary distinction between the two is that the structured road has a driving reference line, and the landscape is more regular. The trajectory does not deviate significantly from the reference line and its curvature continuity is more valued. Compared to driving on a structured road, parking in unstructured environments may necessitate multiple maneuvers and is more difficult in complex and variable environments. The focus of this article is parking trajectory planning in unstructured environments.

(a) (b)

Fig. 1. Unstructured, complex parking scene. (a) Very complex environments with narrow passages formed by stationary vehicles or other obstacles. (b) Deceleration strip at the entrance or exit of the parking lot.

Typical unstructured parking scenarios are shown in Fig. 1, in which no reference line guides the vehicle park. When a complex environment with narrow passages (Fig. 1(a)) is formed by stationary vehicles or other irregular obstacles, it may have a high computing costs and even fail to find the proper initial guess for the subsequent optimization process. In some potentially hazardous local areas (e.g., the entrance or exit of a parking lot), there may be deceleration strips (Fig. 1(b)) that restrict the speed of vehicles in the local area to reduce the safety risk. The global and local speed restrictions are distinct, and

Manuscript received 21 February 2023; revised 21 March 2023 and 7 April 2023; accepted 9 April 2023. Date of publication 18 April 2023; date of current version 20 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61976039 and 52172382, in part by the China Fundamental Research Funds for the Central Universities under Grant DUT22JC09, and in part by the Science and Technology Innovation Fund of Dalian under Grant 2021JJ12GX015. *(Corresponding author: Linhui Li.)*

Jing Lian and Linhui Li are with the School of Automotive Engineering, Faculty of Vehicle Engineering and Mechanics, State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian 116024, China (e-mail: lianjing@dlut.edu.cn; lilinhui@dlut.edu.cn).

Weiwei Ren and Fengning Yu are with the School of Automotive Engineering, Dalian University of Technology, Dalian 116024, China (e-mail: weiweiren@mail.dlut.edu.cn; yfn19941208@mail.dlut.edu.cn).

Dongfang Yang is with the Chongqing Chang'an Automobile Co., Ltd., Chongqing 400023, China (e-mail: yangdf@changan.com.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2023.3268088.

Digital Object Identifier 10.1109/TIV.2023.3268088

the local restrictions are only activated when the vehicle enters a particular area. This "if-else" structure makes optimization problem challenging. Although there are numerous methods for planning parking trajectories in unstructured environments, few can effectively address the aforementioned issues. Our work focuses primarily on narrow passages and "if-else" constraint structures in complex unstructured parking scenes.

B. Related Works

This section presents the methods for autonomous valet parking trajectory planning in unstructured environments. These methods primarily fall into search-and-sample-based and optimization-based [9].

The search-and-sample-based methods consist of two components: the sample in state and control space. They discretize the continuous state or control space into finite fundamental elements. On this basis, the state or control space is converted into a connected graph with nodes and edges, and then graph-search-based methods are used to generate the path between the initial and final nodes. The A* algorithm [10] is the typical method of sampling search in the state space. In addition, researchers have proposed numerous algorithms based on improved A* [11], [12], [13]. However, these methods disregard the vehicle's ackerman steering characteristics and the vehicle's motion posture. Recent sampling techniques in state space have included deterministic methods represented by state lattice [14] and stochastic methods represented by rapidly-exploring random tree (RRT) series [15], [16], [17]. The state lattice approach samples in a three-dimensional state space comprised of vehicle poses and determines the connection path from each node to all nearby nodes using the optimal control method. On this basis, an offline connection graph containing vast amounts of complex information can be constructed. The RRT series algorithm is efficient and robust in most cases but may fail in complex situations involving narrow passages. This uncertainty does not meet parking stability requirements. The Dynamic Window Approach (DWA) [18] and hybrid A* [19] are typical methods for sampling search in the control space. However, the hybrid A* search algorithm has a low search efficiency in narrow scenes and may not always work.

The core of optimization-based methods is to model planning problems based on continuous variables so that planning tasks can be described in a more intuitive, precise, and uniform manner. It discretizes variables to transform them into nonlinear programming problems (NLP). Common NLP are divided into strict constraint and softening constraint methods. The strict constraint methods require that the resulting solution strictly satisfy all constraints in the NLP problem, typically including sequential quadratic programming (SQP) [20], [21] and interior point algorithms (IPM) [22], [23], [24]. Li and Shao [23] proposed the triangle area method for establishing collision avoidance constraints between vehicles and obstacles. Due to its non-differentiability and non-convexity, this constraint makes finding a solution extremely time-consuming. Zhang [25] designed a symbolic distance function to improve the differentiability. These methods are not as effective as softening constraint methods when solving problems. Dolgov et al. [19] formulate collision avoidance and vehicle kinematic constraints as penalty polynomials and linearly combines them as cost functions. However, because there are no rigid constraints, there is a potential collision risk. Time Elastic Band (TEB) [26] was proposed using an external penalty function to describe vehicle kinematics and collision avoidance constraints. The algorithm has a high solution frequency, but it often produces the phenomenon of crossing obstacles or violating vehicle kinematics in complex environments. Although softening constraint methods are quick, the quality of their solutions is low, and it is frequently challenging to strike a balance between vehicle kinematic constraints and collision avoidance constraints.

Combining sampling-and-search-based and optimization-based methods [27], [28], [29], [30] is the preferred strategy for reducing the solution time of NLP problems and ensuring the quality of the solutions. The sampling-and-search-based methods (e.g., A* [10], hybrid A* [19]) generate a rough initial solution by selecting the homotopy class globally, whereas the optimization-based methods can expedite the optimization process by utilizing the initial solution. Despite using high-quality initial solutions to aid the problem-solving process, NLP retains its large scale and strong non-linearity. Therefore, driving corridor modeling strategies [31], [32], [33], [34] are employed to simplify collision avoidance constraints. Specifically, a corridor can be constructed in the vicinity of the initial solution to isolate the vehicle body from surrounding obstacles completely, thereby replacing the complex collision avoidance constraint with the constraint "vehicles must travel in the corridor". However, these methods are not always effective in certain circumstances. First, the initialization strategy in a complex work environment may significantly increase the search time to generate initial guesses due to narrow space or failure to initialize altogether. Although Sheng et al. [34] propose to use a user-defined parameter to determine whether a node is in a narrow or wide passage to improve search efficiency, this method heavily relies on the shape of the obstacle, which causes the initial search and subsequent optimization process to collapse easily when the obstacle width is small. Second, even if the initialization strategy obtains the proper initial solution, the corridor generation must set the appropriate step size [31]. Due to wasted free space, the optimal solution may be lost if the size is excessive. The small step size may necessitate extensive calculations and cannot match the vehicle's real-time performance. The above-optimization-based methods are not ideal for complex situations involving narrow passages. In this scenario, there may also be some potentially hazardous locations (Fig. 1(b)). According to human driving habits, the local speed is lower than the global speed when the vehicle travels through the area. Hence, the global and local speed constraints are distinct, and the local speed constraint is only activated when the vehicle travels through the local area. The "if-else" structure makes optimization problems challenging, and few of aforementioned methods can handle it.

C. Contributions

In order to speed up the numerical solution, this work must satisfy two conditions: 1) Even in complex environments with narrow passages, initial guesses must be generated efficiently.

2) Driving corridor modeling strategies should be employed to simplify complex collision avoidance constraints. As stated previously, the current corridor generation strategy cannot accommodate high efficiency and quality constraints. In the parking field, the classic hybrid A* algorithm may be ineffective or even fail to search in complex environments with narrow passages. This article proposes a two-stage autonomous valet parking trajectory planning framework to address the aforementioned limitations.

These are the primary contributions of this work: 1) An enhanced hybrid A* algorithm is proposed to efficiently search initial guess, even in complex environments with narrow passages formed by obstacles of various shapes; 2) a two-stage generation strategy of the driving corridor is designed to replace the complex collision-avoidance constraint, which can accommodate both high efficiency and high-quality constraints; 3) a novel approximation formulation for the local state constraints is introduced.

D. Organization

The remaining sections of this article are as follows. Section II presents the trajectory planning problem in the form of an OCP for autonomous valet parking. Our proposed two-stage trajectory planning framework is introduced in Section III. Section IV presents the effectiveness of the experimental testing results. Section V concludes the article with its findings.

II. Problem Formulation

In this section, the task of parking trajectory planning in unstructured scenes is formulated as an optimization problem with a time- and energy-minimizing objective function and four distinct constraints: 1) initial and terminal constraints, 2) kinematic equality and inequality constraints, 3) collision avoidance constraints, and 4) local state constraints. The specifics are listed below.

A. Trajectory Planning Problem Formulation

Generally, the parking trajectory planning task from its start state to its final state can be described as the following standard optimization problem:

$$\min J(\boldsymbol{u}(t), \boldsymbol{x}(t), t_f)$$

$$\text{s.t.} \begin{cases} \boldsymbol{x}(0) = \boldsymbol{x}_{\text{start}}, \boldsymbol{x}(t_f) = \boldsymbol{x}_{\text{final}} & \\ \boldsymbol{u}(0) = \boldsymbol{u}_{\text{start}}, \boldsymbol{u}(t_f) = \boldsymbol{u}_{\text{final}} & \\ \dot{\boldsymbol{x}}(t) = f_{\text{kinematics}}(\boldsymbol{u}(t), \boldsymbol{x}(t)) & t \in [0, t_f] \\ \boldsymbol{x}_{\min} \le \boldsymbol{x}(t) \le \boldsymbol{x}_{\max} & t \in [0, t_f] \\ \boldsymbol{u}_{\min} \le \boldsymbol{u}(t) \le \boldsymbol{u}_{\max} & t \in [0, t_f] \\ g_{\text{collision}}(\boldsymbol{x}(t)) \in \mathcal{O}_{\text{free}} = \mathcal{O} \backslash \mathcal{O}_{\text{obs}} & t \in [0, t_f] \end{cases} \tag{1}$$

where J represents the objective function of the optimization problem to be minimized. $\boldsymbol{x}(t) \in \mathbb{R}^{d_x}$ denotes the vehicle state at time t and the vehicle can be driven by controlling actions $\boldsymbol{u}(t) \in \mathbb{R}^{d_u}$ during the parking time of t_f. $[\boldsymbol{x}_{\text{start}}, \boldsymbol{u}_{\text{start}}]$ and $[\boldsymbol{x}_{\text{final}}, \boldsymbol{u}_{\text{final}}]$ denote the start and final states of $[\boldsymbol{x}(t), \boldsymbol{u}(t)]$, respectively. $[\boldsymbol{x}_{\min}, \boldsymbol{u}_{\min}]$ and $[\boldsymbol{x}_{\max}, \boldsymbol{u}_{\max}]$ signify the lower and upper bounds of $[\boldsymbol{x}(t), \boldsymbol{u}(t)]$, respectively. $\dot{\boldsymbol{x}}(t) = f_{\text{kinematics}}(\boldsymbol{u}(t), \boldsymbol{x}(t))$ describes vehicle kinematic constraints through nonlinear equations. The workspace and obstacle space are defined as $\mathcal{O}$ and $\mathcal{O}_{\text{obs}} \in \mathcal{O}$, respectively. The vehicle performs the parking task in free space $\mathcal{O}_{\text{free}} = \mathcal{O} \backslash \mathcal{O}_{\text{obs}}$ and the mapping from vehicle state to vehicle footprints is defined by function $g_{\text{collision}}(\cdot) \in \mathcal{O}_{\text{free}}$. The following section will provide a more comprehensive description of these symbols.

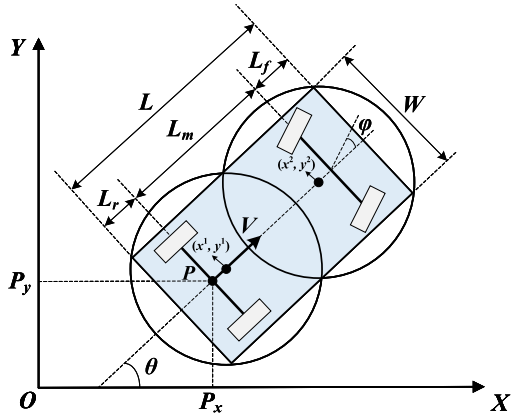


Fig. 2. Geometric parameters related to vehicle kinematics.

B. Constraints

1) Kinematic Equality and Inequality Constraints: This article focuses primarily on the low-speed motion planning of vehicles in parking scenes, so we describe the kinematic equality constraints of vehicles using the classic bicycle model ($\dot{\boldsymbol{x}}(t) = f_{\text{kinematics}}(\boldsymbol{u}(t), \boldsymbol{x}(t))$ in (1)).

$$\begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \dot{\theta}(t) \\ \dot{v}(t) \\ \dot{\varphi}(t) \end{bmatrix} = \begin{bmatrix} v(t) \cdot \cos\theta(t) \\ v(t) \cdot \sin\theta(t) \\ v(t) \cdot \frac{\tan\varphi(t)}{L_m} \\ a(t) \\ \omega(t) \end{bmatrix}, \quad t \in [0, t_f] \tag{2}$$

where $p_x(t)$ and $p_y(t)$ represent the coordinates of the rear axle's center point P. $\theta(t)$, $v(t)$, and $\varphi(t)$ denote the orientation angle, longitudinal velocity, and front-wheel steering angle of the vehicle, respectively. $a(t)$ is defined as acceleration, and $\omega(t)$ stands for angular velocity. So the vehicle state $\boldsymbol{x}(t)$ in (1) can be represented as $[p_x(t), p_y(t), \theta(t), v(t), \varphi(t)]$ and the control state $\boldsymbol{u}(t)$ is defined as $[a(t), \omega(t)]$. In addition to the above parameters, some geometric parameters of the vehicle are also shown in Fig. 2, such as L (vehicle length), L_f (the front overhang length), L_m (vehicle wheelbase), L_r (the rear overhang length) and W (vehicle width).

Generally, discretization is the initial step for numerical evaluation and implementation. The above continuous system model can therefore be discretized as follows:

$$\begin{bmatrix} p_x(k+1) - p_x(k) \\ p_y(k+1) - p_y(k) \\ \theta(k+1) - \theta(k) \\ v(k+1) - v(k) \\ \varphi(k+1) - \varphi(k) \end{bmatrix} = \begin{bmatrix} \frac{t_f}{n} v(k) \cdot \cos\theta(k) \\ \frac{t_f}{n} v(k) \cdot \sin\theta(k) \\ \frac{t_f}{n} v(k) \cdot \frac{\tan\varphi(k)}{L_m} \\ \frac{t_f}{n} a(k) \\ \frac{t_f}{n} \omega(k) \end{bmatrix} \tag{3}$$

where the continuous trajectory $\boldsymbol{x}(t)$ can be discretized into a series of $(n+1)$ waypoints using t_f as the

parking process time, i.e., $[\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(n)]$. $\boldsymbol{x}(k) = [p_x(k), p_y(k), \theta(k), v(k), \varphi(k)]$ denotes the vehicle state at time step $k\frac{t_f}{n}$, $\forall k \in \{0, 1, \ldots, n-1\}$. $\boldsymbol{u}(k) = [a(k), \omega(k)]$ represents the control state at time step $k\frac{t_f}{n}$, $\forall k \in \{0, 1, \ldots, n-1\}$.

The discretized kinematic inequality constraints of the vehicle ($\boldsymbol{x}_{\min} \leq \boldsymbol{x}(t) \leq \boldsymbol{x}_{\max}$ and $\boldsymbol{u}_{\min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{\max}$ in (1)) are as follows:

$$\begin{bmatrix} -a_{\max} \\ -\omega_{\max} \\ -v_{\max} \\ -\varphi_{\max} \end{bmatrix} \leq \begin{bmatrix} a(k) \\ \omega(k) \\ v(k) \\ \varphi(k) \end{bmatrix} \leq \begin{bmatrix} a_{\max} \\ \omega_{\max} \\ v_{\max} \\ \varphi_{\max} \end{bmatrix} \tag{4}$$

where the upper and lower limits are established to constrain the acceleration $a(k)$, angular velocity $\omega(k)$, longitudinal velocity $v(k)$, and front-wheel steering angle $\varphi(k)$ of the vehicle.

2) Initial and Terminal Constraints: The two-point boundary constraints for the initial and terminal states of the vehicle ($\boldsymbol{x}(0) = \boldsymbol{x}_{\text{start}}, \boldsymbol{u}(0) = \boldsymbol{u}_{\text{start}}, \boldsymbol{x}(t_f) = \boldsymbol{x}_{\text{final}}$ and $\boldsymbol{u}(t_f) = \boldsymbol{u}_{\text{final}}$ in (1)) are presented as follows:

$$\begin{bmatrix} x(0) \\ y(0) \\ \theta(0) \\ a(0) \\ \omega(0) \\ v(0) \\ \varphi(0) \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \\ \theta_s \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} x(t_f) \\ y(t_f) \\ \theta(t_f) \\ a(t_f) \\ \omega(t_f) \\ v(t_f) \\ \varphi(t_f) \end{bmatrix} = \begin{bmatrix} x_f \\ y_f \\ \theta_f \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

where $[x_s, y_s, \theta_s]$ and $[x_f, y_f, \theta_f]$ are the start and final vehicle configurations at $t = 0$ and $t = t_f$, respectively. In this study, the vehicle is placed in a stable state before and following the autonomous parking procedure.

3) Collision Avoidance Constraints: The collision-avoidance constraint $g_{\text{collision}}(\boldsymbol{x}(t)) \in \mathcal{O}_{\text{free}} = \mathcal{O} \backslash \mathcal{O}_{\text{obs}}$ in (1) is essential for an autonomous vehicle to avoid collisions with dynamic and static obstacles. Considering that the vehicle is rectangular or when the obstacles are polytopic, the collision-avoidance constraint is generally nondifferentiable and challenging to solve by a gradient-based OCP solver. Some methods of building driving corridors in recent years have been proposed to solve this problem; it separates the ego vehicle from the surrounding obstacles by driving corridors, keeping the vehicle always in the free space of the corridor and replacing the nominal collision-avoidance constraint with driving corridor constraints. Our work follows previous research [31] in that we first search for the initial coarse trajectory to obtain dense discrete points. Then we inflate the rectangular along directions 1, 2, 3, and 4 on each discrete point until it has the greatest possible free area. The driving corridor restrictions can be expressed as follows:

$$\begin{aligned} &\mathrm{x}_{\min}^j \leq x^j(t) \leq \mathrm{x}_{\max}^j, \\ &\mathrm{y}_{\min}^j \leq y^j(t) \leq \mathrm{y}_{\max}^j, \\ &t = k \cdot t_f/n, \forall k \in \{0, 1, \ldots, n-1\}, \forall j \in \{1, \ldots, N_C\} \end{aligned} \tag{6}$$

As shown in Fig. 2, N_c discs are used to cover the rectangular vehicle body and the center coordinate of the jth disk is represented by (x^j, y^j). it can be described by the geometric

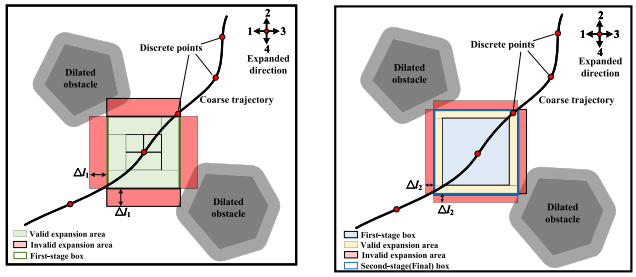


Fig. 3. Two-stage generation strategy of the driving corridor along each discrete point. (a) Generate the first-stage box with a large step size Δl_1 rapidly. (b) Generate a second-stage (final) box with a small step size Δl_2 based on the first-stage box.

dimensions of the vehicle:

$$\begin{aligned} x^j(t) &= p_x(t) + \left(\frac{2j-1}{2N_c} \cdot (L_f + L_m + L_r) - L_r\right) \cdot \cos\theta(t), \\ y^j(t) &= p_y(t) + \left(\frac{2j-1}{2N_c} \cdot (L_f + L_m + L_r) - L_r\right) \cdot \sin\theta(t), \\ &\forall j \in \{1, \ldots, N_c\}, t \in [0, t_f] \end{aligned} \tag{7}$$

where the center coordinate (x^j, y^j) of the jth disc can be determined by the vehicle state $[p_x(t), p_y(t), \theta(t)]$. The radius R_d of each disc can be calculated as follows:

$$R_d = \frac{1}{2}\sqrt{\left(\frac{L_f + L_m + L_r}{N_c}\right)^2 + W^2} \tag{8}$$

To prevent the vehicle from colliding with obstacles, the distance between the center of each disc and the obstacles must be at least equal to the radius R_d. Consequently, each disc can be shrunk to its center, and the obstacles can be enlarged by R_d. Due to space limitations, readers can refer to [31] for a comprehensive explanation of the corridor generation procedure. The expansion strategy, however, is contingent upon selecting an appropriate step size. The optimal solution may be lost due to wasted free space when the step size is large. The small step size may necessitate much calculation in specific environments and is incompatible with the vehicle's real-time performance. To address this issue, we propose a two-stage driving corridor generation strategy along each discrete point, as depicted in Fig. 3. The first-stage box is generated rapidly with a significant expansion step size Δl_1, and then based on the first-stage box, the second-stage (final) box is generated with a small step size Δl_2. The enhancements have significantly increased the effectiveness of corridor generation without sacrificing free space.

4) Local State Constraints: When a vehicle enters a specific zone (e.g., deceleration strips at the entrance or exit of the parking lot), the local state constraints are activated. According to people's driving experience, the vehicle's speed must be reduced in these areas to prevent danger or vehicle damage. This constraint differs from the preceding in that it requires the evaluation of the vehicle's state in various regions to be coupled with the constraints of the vehicle's state; this makes the problem

challenging. Inspired by [35], the following novel approximate formula is intended to solve this problem:

$$f_{local}\left(p_x(k), p_y(k)\right) \cdot f_{state}(\mathcal{Z}(k)) = 0 \tag{9}$$

where $f_{local}(p_x(k), p_y(k))$ describes the vehicle is located in the local area, $f_{state}(\mathcal{Z}(k))$ represents the constraint on the desired vehicle state variable $\mathcal{Z}(k)$. It should also be noted that when the vehicle is in the local area, $f_{local}(p_x(k), p_y(k))$ is defined as non-zero and $f_{state}(\mathcal{Z}(k))$ must be zero due to (9). Similarly, when the vehicle is in other areas, $f_{local}(p_x(k), p_y(k))$ must be zero and $f_{state}(\mathcal{Z}(k))$ is not constrained. To meet the previous requirements, we follow the strategy in [35] to define $f_{local}(p_x(k), p_y(k))$ as follows:

$$\begin{aligned} f_{\text{local}}\left(p_x(k), p_y(k)\right) &= f_x\left(p_x(k)\right) \cdot f_y\left(p_y(k)\right) \\ &= \max\left(-\left(p_x(k) - x_a\right)\left(p_x(k) - x_b\right), 0\right) \\ &\quad \cdot \max\left(-\left(p_y(k) - y_a\right)\left(p_y(k) - y_b\right), 0\right) \end{aligned} \tag{10}$$

where the local area of the deceleration strip is simplified into a rectangle. Its area is defined as $x_a \le p_x(k) \le x_b$ and $y_a \le p_y(k) \le y_b$. As for $f_{state}(\mathcal{Z}(k))$, a novel formula is designed to constrain vehicle state in the local area as follows:

$$f_{state}(\mathcal{Z}(k)) = \max\left(\left(\mathcal{Z}(k) - \mathcal{Z}_a\right)\left(\mathcal{Z}(k) - \mathcal{Z}_b\right), 0\right) \tag{11}$$

Therefore, (9) can be expressed by the following formula:

$$\begin{aligned} J_{\max}(\boldsymbol{x}(k)) = &\max\left(-\left(p_x(k) - x_a\right)\left(p_x(k) - x_b\right), 0\right) \\ &\cdot \max\left(-\left(p_y(k) - y_a\right)\left(p_y(k) - y_b\right), 0\right) \\ &\cdot \max\left(\left(\mathcal{Z}(k) - \mathcal{Z}_a\right)\left(\mathcal{Z}(k) - \mathcal{Z}_b\right), 0\right) \end{aligned} \tag{12}$$

Given that the maximum function is not differentiable, it is challenging to solve the optimization problem. The maximum function is approximated and smoothed using the Log-SumExp (LSE) function. Assume that $x = (x_1, x_2, \ldots, x_n)$, $x_{\max} = \max(x_1, x_2, \ldots, x_n)$, LSE can be defined as:

$$\mathrm{LSE}(\beta x) = \log \sum_{i=1}^{n} e^{\beta x_i} \tag{13}$$

and

$$x_{\max} \approx \frac{1}{\beta}\,\mathrm{LSE}(\beta x) \tag{14}$$

where we can scale the function by adjusting the parameter β to tighten the bounds, then the smoothed function of (12) can be defined as:

$$\begin{aligned} J_{\log}(\boldsymbol{x}(k)) = \frac{1}{\beta}\Big[&\log\left(e^{-\beta\left(p_x(k) - x_a\right)\left(p_x(k) - x_b\right)} + 1\right) \\ &\cdot \log\left(e^{-\beta\left(p_y(k) - y_a\right)\left(p_y(k) - y_b\right)} + 1\right) \\ &\cdot \log\left(e^{\beta\left(\mathcal{Z}(k) - \mathcal{Z}_a\right)\left(\mathcal{Z}(k) - \mathcal{Z}_b\right)} + 1\right)\Big] = 0 \end{aligned} \tag{15}$$

C. Objective Function

The objective function $J(\boldsymbol{u}(t), \boldsymbol{x}(t), t_f)$ in (1) is defined as follows:

$$\begin{aligned} \min J = \mu_1 t_f + \mu_2 \sum_{k=0}^{n-1} \|v(k+1) - v(k)\|_2^2 \\ + \mu_3 \sum_{k=0}^{n-1} \|\varphi(k+1) - \varphi(k)\|_2^2 \end{aligned} \tag{16}$$

where $\mu_1 > 0$, $\mu_2 > 0$ and $\mu_3 > 0$ denote the weighting parameters. In this article, time t_f is penalized for the minimum parking time, the acceleration $\|v(k+1) - v(k)\|_2^2$ and angular velocity $\|\varphi(k+1) - \varphi(k)\|_2^2$ are penalized for the smoothness of the parking trajectory.

As a summary of the entire section, the autonomous valet parking trajectory planning problem can be stated as the following OCP:

$$\min \text{objective function (16)}$$

$$\text{s.t.} \begin{cases} \text{Kinematic constraints (3), (4) and (7)} \\ \text{Initial and terminal constraints (5)} \\ \text{Within-driving corridor constraints (6)} \\ \text{Local state constraints (15)} \end{cases} \tag{17}$$

III. Proposed Trajectory Planner

This section proposes a two-stage autonomous valet parking trajectory planning framework, and Algorithm 1 introduces the pseudo-codes.

The inputs consist of the information of the start and goal nodes, the information on the map, including obstacles and vehicles, and the output is an optimized trajectory $\mathcal{X}_0$. At the beginning, the function InitializeParameters() in line 1 is used to set some fundamental parameters (e.g., vehicle geometry parameters, basic parameters of search algorithms, and so on). Then, to simplify collision detection, CreateDilatedMap(map) is used in line 2 of Algorithm 1 to expand the obstacles on the grid map with the radius R_d (8). Each disc covering the vehicle is shrunk to a mass point. The algorithm consists of a total of two stages. In order to efficiently and swiftly search for a high-quality initial guess in complex scenes, the first stage provides an enhanced hybrid A* algorithm for dealing with the narrow passages formed by obstacles of various shapes. The second stage employs the primal-dual interior-point solver IPOPT [36] to solve an OCP for a comfortable, smooth, and feasible trajectory. The specifics are listed below.

A. First Stage: Search a Coarse Trajectory to the OCP

Based on the A* algorithm, the classic hybrid A* algorithm adds constraints on vehicle steering, and the search dimension changes from 2D to 3D. Appropriate warm-starting is advantageous to the numerical optimization procedure. Still, in highly complex environments with narrow passages, it may incur a high computational cost or fail to find the proper initial guess. To address these difficulties, an enhanced hybrid A* algorithm

Algorithm 1: Trajectory Planning for Autonomous Parking.

Input: The information of map, start and goal nodes
Output: An optimized trajectory $\mathcal{X}_0$

1: InitializeParameters();
2: map $\leftarrow$ CreateDilatedMap(map);
3: First stage: search a coarse trajectory to the OCP
4: $\mathcal{I}_{\text{wide}} \leftarrow 1, \mathcal{I}_{\text{narrow}} \leftarrow 0, \mathcal{X}_0 \leftarrow \emptyset, \mathcal{L}_{\text{thre}} \leftarrow 5$;
5: WidePathsSet $\mathcal{S}_{\text{WPS}} \leftarrow \emptyset$, NarrowPathsSet $\mathcal{S}_{\text{NPS}} \leftarrow \emptyset$;
6: WideNodesSet $\mathcal{S}_{\text{WNS}} \leftarrow \emptyset$, NarrowNodesSet $\mathcal{S}_{\text{NNS}} \leftarrow \emptyset$;
7: $\mathcal{X}_{\text{AStar}}(x_{\text{AStar}}, y_{\text{AStar}}) \leftarrow$ SearchAStarPath(map);
8: $\mathcal{T}_1 \leftarrow$ GenerateFirstStageCor($\mathcal{X}_{\text{AStar}}, \Delta l_1$);
9: $\mathcal{T}_2 \leftarrow$ GenerateSecondStageCor($\mathcal{T}_1, \Delta l_2$);
10: **for** each node $\mathcal{X}_{\text{node}}$ in $\mathcal{X}_{\text{AStar}}$, **do**
11: $\mathcal{I}_{\text{node}} \leftarrow$ JudgeNode($\mathcal{T}_2, \mathcal{X}_{\text{node}}, \mathcal{L}_{\text{thre}}$);
12: **if** ($\mathcal{I}_{\text{node}} = \mathcal{I}_{\text{wide}}$) **then**
13: push $\mathcal{X}_{\text{node}}$ into $\mathcal{S}_{\text{WNS}}$;
14: **if** ($\mathcal{S}_{\text{NNS}} \neq \emptyset$) **then**
15: push $\mathcal{S}_{\text{NNS}}$ into $\mathcal{S}_{\text{NPS}}$;
16: $\mathcal{S}_{\text{NNS}} \leftarrow \emptyset$;
17: **end if**
18: **else if** ($\mathcal{I}_{\text{node}} = \mathcal{I}_{\text{narrow}}$) **then**
19: push $\mathcal{X}_{\text{node}}$ into $\mathcal{S}_{\text{NNS}}$;
20: **if** ($\mathcal{S}_{\text{WNS}} \neq \emptyset$) **then**
21: push $\mathcal{S}_{\text{WNS}}$ into $\mathcal{S}_{\text{WPS}}$;
22: $\mathcal{S}_{\text{WNS}} \leftarrow \emptyset$;
23: **end if**
24: **end if**
25: **end for**
26: **if** ($\mathcal{S}_{\text{WNS}} \neq \emptyset$) **then**
27: push $\mathcal{S}_{\text{WNS}}$ into $\mathcal{S}_{\text{WPS}}$;
28: **else if** ($\mathcal{S}_{\text{NNS}} \neq \emptyset$) **then**
29: push $\mathcal{S}_{\text{NNS}}$ into $\mathcal{S}_{\text{NPS}}$;
30: **end if**
31: $\mathcal{X}_{\text{seq}} \leftarrow$ IntegrateWideNarrowPath($\mathcal{S}_{\text{WPS}}, \mathcal{S}_{\text{NPS}}$);
32: $\mathcal{X}_{\text{bou}} \leftarrow$ ExtractBoundaryPointsOfEachPath($\mathcal{X}_{\text{seq}}$);
33: $\mathcal{X}^1_{\text{bou}} \leftarrow$ CorrectBoundaryPoints($\mathcal{X}_{\text{bou}}$);
34: $\mathcal{X}^{3\,d}_{\text{bou}} \leftarrow$ SetInitialOrientationAngle ($\mathcal{X}^1_{\text{bou}}, \mathcal{X}_{\text{seq}}$);
35: $\mathcal{X}_0 \leftarrow$ SearchHybridAStarEachAdjacentNode($\mathcal{X}^{3\,d}_{\text{bou}}$);
36: $\mathcal{V}_0 \leftarrow$ GenerateInitialSolutionGuess($\mathcal{X}_0$);
37: Second stage: optimize the coarse trajectory
38: $\mathcal{N}_{\text{iter}} \leftarrow 0, \mathcal{N}_{\max} \leftarrow 10, \mathcal{W}_{\text{penalty}} \leftarrow 10^6, \mathcal{J}_{\text{inf}} \leftarrow +\infty$;
39: **while** ($\mathcal{J}_{\text{inf}} > \mathcal{E}_{\text{tol}}$), **do**
40: $\mathcal{T}_3 \leftarrow$ GenerateFirstStageCor($\mathcal{X}_0, \Delta l_1$);
41: $\mathcal{T}_4 \leftarrow$ GenerateSecondStageCor($\mathcal{T}_3, \Delta l_2$);
42: $\mathcal{S}_{\text{ocp}} \leftarrow$ FormulateOCP($\mathcal{T}_4$);
43: $\mathcal{V}_0 \leftarrow$ SolveOptimizationProblem($\mathcal{S}_{\text{ocp}}, \mathcal{V}_0$);
44: **if** ($\mathcal{N}_{\text{iter}} > \mathcal{N}_{\max}$) **then**
45: **return** failure;
46: **end if**;
47: $\mathcal{J}_{\text{inf}} \leftarrow$ MeasureInfeasibility($\mathcal{V}_0$);
48: $\mathcal{X}_0 \leftarrow$ ExtractOptimalTrajectory($\mathcal{V}_0$);
49: $\mathcal{N}_{\text{iter}} \leftarrow \mathcal{N}_{\text{iter}}$+1;
50: $\mathcal{W}_{\text{penalty}} \leftarrow c\mathcal{W}_{\text{penalty}}$;
51: **end while**
52: **return** An optimized trajectory $\mathcal{X}_0$

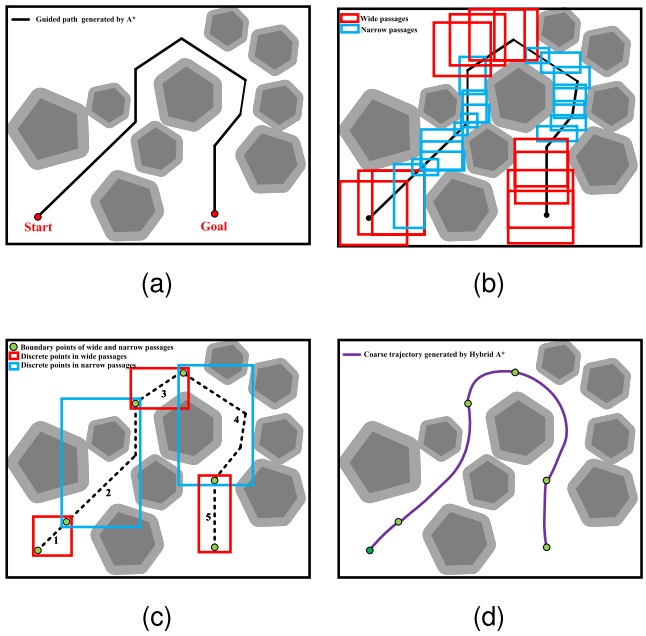


Fig. 4. Searching a coarse trajectory with an enhanced hybrid A*. (a) Fast search for a two-dimensional path by traditional A* algorithm. (b) Laying a driving corridor along the path of (a), and judging whether each node is located in a wide or narrow channel by the length or width of the box. (c) Extracting boundary points of wide and narrow passages. (d) Fast search by hybrid A* algorithm between each adjacent node in (c).

is proposed. As shown in Fig. 4, the EHA process can be broken down into four distinct phases.

Phase 1: In line 7 of Algorithm 1, SearchAStarPath(map) is used to generate a two-dimensional path $\mathcal{X}_{\text{AStar}}$ ($x_{\text{AStar}}, y_{\text{AStar}}$) rapidly using traditional A* algorithm (Fig. 4(a));

Phase 2: After acquiring the two-dimensional path $\mathcal{X}_{\text{AStar}}$, a driving corridor is generated along these discrete points (Fig. 4(b)). The function of GenerateFirstStageCor($\mathcal{X}_{\text{AStar}}, \Delta l_1$) in line 8 of Algorithm 1 is used to generate the first-stage box with a large step size Δl_1 (Fig. 3(a)), and then GenerateSecondStageCor($\mathcal{T}_1, \Delta l_2$) in line 9 denotes the second-stage (final) box with a small step size Δl_2 based on the first-stage box (Fig. 3(b)). The four side lengths of the ith box are represented by $\mathcal{T}_2$ ($\mathcal{L}^i_{\text{up}}$, $\mathcal{L}^i_{\text{left}}$, $\mathcal{L}^i_{\text{down}}$, $\mathcal{L}^i_{\text{right}}$). The two-stage generation strategy not only increases efficiency significantly but also disregards the effect of obstacle shapes when determining whether a node is in wide or narrow passages. By comparing the minimum side length with the threshold $\mathcal{L}_{\text{thre}}$, we can determine whether each node $\mathcal{X}_{\text{node}}$ is in a wide or a narrow passage by JudgeNode ($\mathcal{T}_2, \mathcal{X}_{\text{node}}, \mathcal{L}_{\text{thre}}$) in line 11. If the result $\mathcal{I}_{\text{node}}$ equals $\mathcal{I}_{\text{wide}}$, the node is considered to be in the wide passage and we push the node $\mathcal{X}_{\text{node}}$ into a wide node set $\mathcal{S}_{\text{WNS}}$, several consecutive nodes form a wide path. Similarly, it will also form a narrow node set $\mathcal{S}_{\text{NNS}}$ in a narrow passage. Notably, when the "for" loop goes from a wide node to a narrow node, if the $\mathcal{S}_{\text{WNS}}$ is not empty, it should be pushed into a wide path set $\mathcal{S}_{\text{WPS}}$ which is defined as the cell array in matlab. After that, $\mathcal{S}_{\text{WNS}}$ needs to be set to empty. We can perform similar operations on $\mathcal{S}_{\text{NNS}}$ and the narrow path set $\mathcal{S}_{\text{NPS}}$. Finally, $\mathcal{S}_{\text{WPS}}$ includes one or more paths in wide passages (e.g., paths 1, 3 and 5 in Fig. 4(c).) whereas

$\mathcal{S}_{\text{NPS}}$ comprises of one or more paths in narrow passages (e.g., paths 2, 4 in Fig. 4(c).).

Phase 3: The function IntegrateWideNarrowPath($\mathcal{S}_{\text{WPS}}, \mathcal{S}_{\text{NPS}}$) in line 31 of Algorithm 1 is used to integrate multiple wide and narrow paths in $\mathcal{S}_{\text{WPS}}$ and $\mathcal{S}_{\text{NPS}}$ (e.g., as shown in Fig. 4(c), $\mathcal{S}_{\text{WPS}}$ and $\mathcal{S}_{\text{NPS}}$ are integrated into $\mathcal{X}_{\text{seq}}$ including multiple paths in the order of 1, 2, 3, 4, and 5.). Then, the function ExtractBoundaryPointsOfEachPath($\mathcal{X}_{\text{seq}}$) in line 32 is deployed to extract the boundary nodes $\mathcal{X}_{\text{bou}}$ (e.g., green nodes in Fig. 4.) of each path and CorrectBoundaryPoints($\mathcal{X}_{\text{bou}}$) is used to select the appropriate boundary point by threshold $\mathcal{L}_{\text{Pthre}}$. However, it only contains location data (p_x, p_y). The start and goal nodes include the orientation angle information. We need to initialize the orientation angles for other boundary nodes. Consequently, the orientation angle of the final node in dth path can be initialized using the following formula:

$$\theta_{init}^{d} = arctan\left(\frac{p_{yend}^{d} - p_{yend-1}^{d}}{p_{xend}^{d} - p_{xend-1}^{d}}\right) \tag{18}$$

The function SetInitialOrientationAngle ($\mathcal{X}_{\text{bou}}^{1}$, $\mathcal{X}_{\text{seq}}$) in line 34 is used to implement the aforementioned procedure and obtain the 3D boundary nodes $\mathcal{X}_{\text{bou}}^{3\,d}$ $(p_x, p_y, \theta_{node})$.

Phase 4: The hybrid A* is adopted between each adjacent node and the coarse trajectory $\mathcal{X}_0$ is obtained (Fig. 4(d)) by SearchHybridAStarEachAdjacentNode($\mathcal{X}_{\text{bou}}^{3\,d}$) on line 35. In line 36, GenerateInitialSolutionGuess($\mathcal{X}_0$) is mainly getting other all decision variables on the coarse trajectory $\mathcal{X}_0$ (p_x^0, p_y^0, θ^0), and returning the final initial trajectory guess $\mathcal{V}_0$ (p_x^0, p_y^0, θ^0, v^0, φ^0, a^0, w^0, t_f^0). The t_f^0 is obtained according to configuration space, $v_{\max}$ and $a_{\max}$. The states v^0, φ^0, a^0, w^0 can be set as follows, where $v^0(n)$, $\varphi^0(n)$ are user-defined:

$$v^0(k) = \frac{\left\|P^0(k+1) - P^0(k)\right\|_2}{t_f^0/n}, \forall k \in \{0, \ldots, n-1\} \tag{19}$$

$$\varphi^0(k) = \tan^{-1}\left(\frac{\theta^0(k+1) - \theta^0(k)}{t_f^0/n}\frac{L_m}{v^0(k)}\right) \quad \forall k \in \{0, \ldots, n-1\} \tag{20}$$

$$w^0(k) = \frac{\varphi^0(k+1) - \varphi^0(k)}{t_f^0/n}, \forall k \in \{0, \ldots, n-1\} \tag{21}$$

$$a^0(k) = \frac{v^0(k+1) - v^0(k)}{t_f^0/n}, \forall k \in \{0, \ldots, n-1\} \tag{22}$$

B. Second Stage: Optimize the Coarse Trajectory

In the second stage, the initial guess $\mathcal{V}_0$ obtained in the first stage is used to warm-start the numerical optimization process. This phase optimizes the trajectory through the construction of an iterative framework. Since the driving corridor is constructed based on the low-quality initial solution $\mathcal{V}_0$ generated by the first stage, some free space may be lost. Therefore, the direct solution to (17) may not always work. In order to address the aforementioned issue, a while loop is created and iterated numerous times. An intermediate OCP problem is constructed and solved during each iteration. A new OCP is built in the next iteration by regenerating the driving corridor based on the previous iteration's optimal solution. Through this iterative framework, we hope to maximize the use of free space and improve solution quality. This is accomplished by simplifying (17) as the aforementioned intermediate OCP (23), which is constructed by softening the nonlinear kinematics-related equality constraints (3), (7) and local state equality constraint (15) as external penalty costs and merging them into the cost function:

$$\begin{aligned} &\min\ (16) + \mathcal{W}_{\text{penalty}} \cdot \left(J_{\text{penalty }(3)} + J_{\text{penalty }(7)} + J_{\text{penalty }(15)}\right) \\ &\text{s.t.} \begin{cases} \text{Kinematic constraints (4)} \\ \text{Initial and terminal constraints (5)} \\ \text{within-driving corridor constraints (6)} \end{cases} \end{aligned} \tag{23}$$

where $\mathcal{W}_{\text{penalty}}$ represents the weighting parameter. $J_{\text{penalty }(3)}$, $J_{\text{penalty }(7)}$ and $J_{\text{penalty }(15)}$ denote the penalty costs of (3), (7), and (15), respectively. Thus, for example, $J_{\text{penalty }(3)}$ can be expressed as the following:

$$\begin{aligned} J_{\text{penalty }(3)} = &\sum_{k=0}^{n-1}\left\|p_x(k+1) - p_x(k) - \frac{t_f}{n}v(k)\cos\theta(k)\right\|^2 \\ &+ \sum_{k=0}^{n-1}\left\|p_y(k+1) - p_y(k) - \frac{t_f}{n}v(k)\sin\theta(k)\right\|^2 \\ &+ \sum_{k=0}^{n-1}\left\|\theta(k+1) - \theta(k) - \frac{t_f}{n}v(k)\frac{\tan\varphi(k)}{L_m}\right\|^2 \\ &+ \sum_{k=0}^{n-1}\left\|v(k+1) - v(k) - \frac{t_f}{n}a(k)\right\|^2 \\ &+ \sum_{k=0}^{n-1}\left\|\varphi(k+1) - \varphi(k) - \frac{t_f}{n}\omega(k)\right\|^2 \end{aligned} \tag{24}$$

Lines 37–52 of Algorithm 1 present the pseudo-codes for the second stage. In line 38, several fundamental parameters are initialized (e.g., current iteration times $\mathcal{N}_{\text{iter}}$, maximum iterations times $\mathcal{N}_{\max}$, weighting parameter $\mathcal{W}_{\text{penalty}}$ and infeasibility degree $\mathcal{J}_{\text{inf}}$ representing the sum of $J_{\text{penalty }(3)}$, $J_{\text{penalty }(7)}$ and $J_{\text{penalty }(15)}$). In line 39, $\mathcal{E}_{\text{tol}}$ represents the user-defined convergence threshold and the softened nonlinear kinematics-related and local state constraints are satisfied if $\mathcal{J}_{\text{inf}} < \mathcal{E}_{\text{tol}}$. Similarly, if $\mathcal{N}_{\text{iter}}$ is greater than the maximum iterations times $\mathcal{N}_{\max}$, then the solution will fail. GenerateFirstStageCor($\mathcal{X}_0, \Delta l_1$) and GenerateSecondStageCor($\mathcal{T}_3, \Delta l_2$) are used to construct new driving corridors during each iteration. The intermediate OCP (23) is constructed by FormulateOCP($\mathcal{T}_4$) in line 42. Based on the warm-starting initial guess $\mathcal{V}_0$, SolveOptimizationProblem($\mathcal{S}_{\text{ocp}}$, $\mathcal{V}_0$) is used to solve (23) by primal-dual interior-point solver IPOPT [36]. The infeasibility degree $\mathcal{J}_{\text{inf}}$ can be calculated by MeasureInfeasibility($\mathcal{V}_0$) in line 47. ExtractOptimalTrajectory($\mathcal{V}_0$) in line 48 is used to extract the optimal trajectory $\mathcal{X}_0$ from the solution vector, and $\mathcal{X}_0$ is considered the benchmark for the subsequent iterative process to construct the corridor until $\mathcal{J}_{\text{inf}} < \mathcal{E}_{\text{tol}}$.

TABLE I
BASIC PARAMETERS

Symbol	Description	Value
L	Vehicle length	$4.8m$
L_f	Front overhang length	$0.95m$
L_m	Vehicle wheelbase	$2.8m$
L_r	Rear overhang length	$1.05m$
W	Vehicle width	$1.9m$
$a_{\max}$	Maximum acceleration	$2m/s^2$
$\omega_{\max}$	Maximum angular velocity	$0.85rad/s$
$v_{\max}$	Maximum velocity	$5m/s$
$\varphi_{\max}$	Maximum steering angle	$0.85rad$
μ_1, μ_2, μ_3	Weighting parameters of (16)	1, 0.01, 0.01
$\mathcal{W}_{\text{penalty}}$	Initial weight in (23)	10^4
n	Number of sampled finite elements	200
N_c	Number of discs covering the vehicle body	2
$\mathcal{N}_{\max}$	Maximum iterations of Alg. 1	10
c	Growth coefficient for $\mathcal{W}_{\text{penalty}}$	5
$\mathcal{E}_{\text{tol}}$	Convergence threshold of Alg. 1	10^{-4}
Δl_1, Δl_2	Expansion step size of first/second-stage box	1m, 0.1m
l_{max}	Maximum side length of the box	8m
$\mathcal{L}_{\text{Pthre}}$	Threshold for selecting boundary point	30
$\mathcal{L}_{\text{thre}}$	Threshold for determining wide/narrow passages	4.5m
Δd_x, Δd_y	The xoy resolution for the IHA	$0.2m$, $0.2m$
Δd_{theta}	The heading resolution for the IHA	0.2
β	The parameter scaling the smoothed function (15)	10

IV. EXPERIMENTAL TESTING RESULTS

A. Implementation Details

The simulation experiment was conducted using Matlab R2021a (Win64) and a computer with an Intel Core i9-10850 K CPU@3.60 GHz and 32 GB RAM. The OCP is solved by the primal-dual interior-point solver IPOPT [36]. Some detailed basic parameters are shown in Table I.

B. Analysis of the Proposed Planner

This subsection demonstrates the performance of our proposed method in various environments containing arbitrarily shaped obstacles.

We evaluated the proposed planner in the four scenarios depicted in Fig. 5: Fig. 5(a) depicts a scene with randomly placed obstacles, Fig. 5(b) depicts a scene in a familiar parking lot, Fig. 5(c) depicts a narrow passage scene formed by obstacles of varying shapes, and Fig. 5(d) depicts a narrow passage scene formed by common, slender obstacles. In order to test the adaptability of the trajectory planning algorithm to various scenarios, the local state constraints (15) are removed temporarily. The optimized trajectories and vehicle footprints demonstrate that our method can successfully generate a comfortable, feasible, and smooth trajectory in both simple and complex scenarios. Due to page limitation, we will only use Fig. 5(c) to illustrate the generation process of the optimized trajectory, as shown in Fig. 6. The pink line in Fig. 6(a) represents the lead lines searched by A*. The red and green boxes in Fig. 6(b) indicate wide and narrow passages, respectively. The boundary points of each passage are depicted in Fig. 6(c), and we improve search

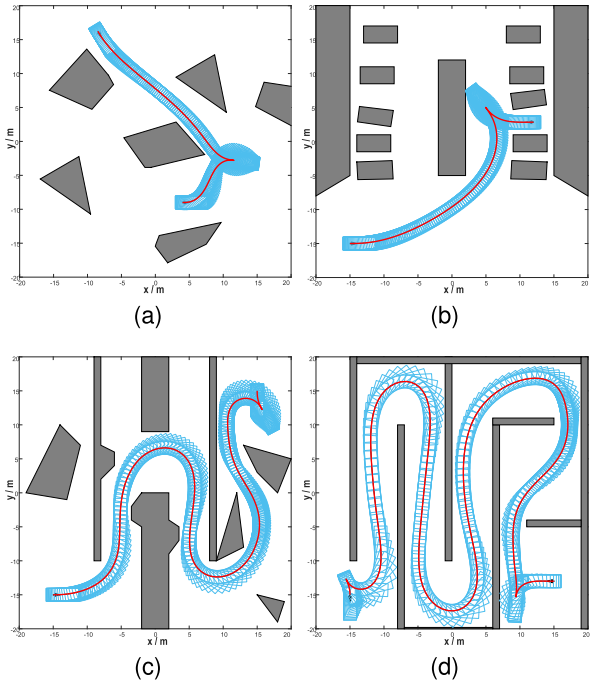


Fig. 5. Simulation results for four different scenarios utilizing our proposed planner. (a) The scene with randomly placed obstacles. (b) The scene in familiar parking lots. (c) The scene involving a narrow passage formed by obstacles of varying shapes. (d) The scene involving a narrow passage formed by obstacles of regular slender.

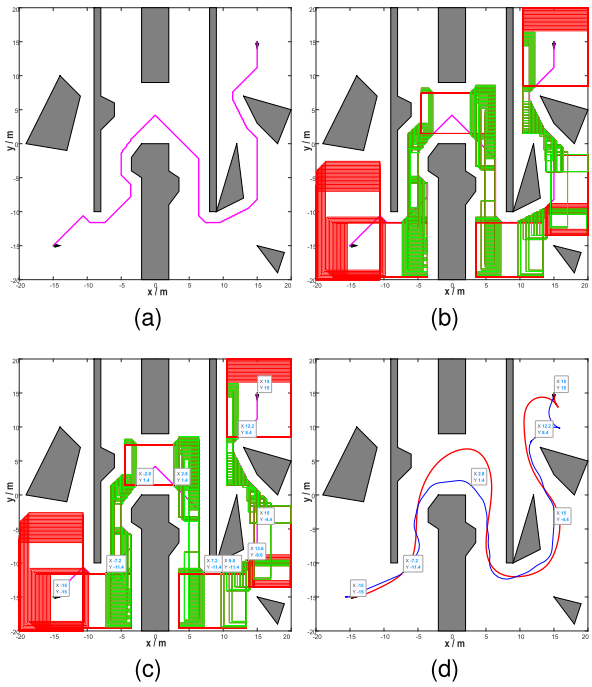


Fig. 6. The generation process of the optimized trajectory in Fig. 5(c).

efficiency by setting the threshold $\mathcal{L}_{\text{Pthre}}$. When the number of boxes in a passage falls below this threshold, the left passage boundary will be removed. This operation will significantly increase our method's flexibility. The number of boundary points decreases from 10 in Fig. 6(c) to 6 in Fig. 6(d). The boundary

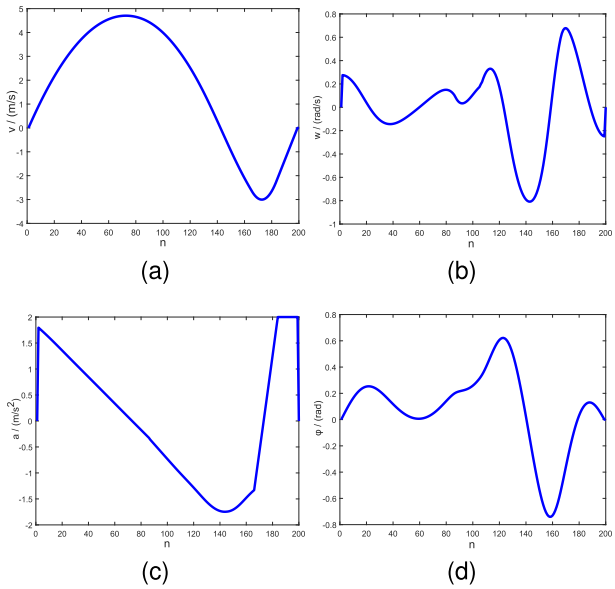


Fig. 7. The velocity v, angular velocity w, acceleration a, and steering angle φ of the optimized trajectory in Fig. 5(b).

TABLE II
PERFORMANCE COMPARISON

Scene	Method	CPU time I	CPU time II	Total time
Scene a	A*-IPOPT	0.047	1.679	1.726
	Hybrid A*-IPOPT	3.130	0.378	3.508
	FTHA-IPOPT	3.101	0.389	3.490
	OURS	0.254	0.556	0.810
Scene b	A*-IPOPT	0.143	Failed	Failed
	Hybrid A*-IPOPT	11.916	0.383	12.299
	FTHA-IPOPT	11.624	0.369	11.993
	OURS	1.793	0.339	2.132
Scene c	A*-IPOPT	0.651	Failed	Failed
	Hybrid A*-IPOPT	Failed	Failed	Failed
	FTHA-IPOPT	16.275	Failed	Failed
	OURS	0.933	0.366	1.299
Scene d	A*-IPOPT	0.984	Failed	Failed
	Hybrid A*-IPOPT	Failed	Failed	Failed
	FTHA-IPOPT	21.345	Failed	Failed
	OURS	1.387	0.460	1.847

points are then connected to produce the blue line depicted in Fig. 6(d) as the initial guess, while the red line represents the optimal smooth trajectory. In addition, Fig. 7 illustrates the velocity v, angular velocity w, acceleration a, and steering angle φ of the optimized trajectory in Fig. 5(b). It can be seen that the values satisfy the requirements of Table I and maintain a degree of smoothness.

C. Comparisons With Prevalent Maneuver Planners

In this section, a comparison is made between the proposed two-stage autonomous valet parking trajectory planning framework and prevalent existing planners.

As shown in Table II, the algorithms being compared are A*-IPOPT, Hybrid A*-IPOPT, FTHA-IPOPT and our own. The A*-IPOPT denotes that the initial guess is searched using A* algorithm, and the OCP is solved using the IPOPT solver. The Hybrid A*-IPOPT is a parking planning method that combines search and optimal control. It obtains the initial trajectory satisfying the vehicle's kinematics via hybrid A* and employs IPOPT for numerical optimization. The FTHA-IPOPT is a fault-tolerant hybrid A* algorithm that combines the benefits of hybrid A* and A*, ensuring that even if hybrid A* fails to search in narrow passages, the path from the current node to the target node can be searched using A*. In order to compare the performance of these algorithms in these four scenarios (Fig. 5), we refer to the time spent searching for the initial guess as CPU time I and the time required to solve the numerical optimization process as CPU time II. The final test results for efficiency are displayed in Table II.

In these four scenarios, the A*-IPOPT is more efficient in searching for initial guesses; however, due to the low quality of the initial solution, it is easy to consume more time (Scene a) or even be unable to find the optimal solution (Scenes b, c, and d) when solving the optimization problem. In simple scenarios (Scenes a and b), the Hybrid A*-IPOPT can slowly find feasible solutions and optimize them. However, in certain complex scenarios, it may fail to find the proper initial guess for the subsequent optimization process (Scenes c and d). Although the FTHA-IPOPT can guarantee a successful initial guess in these four situations, it takes more time (Scenes a and b) and may fail to find the optimal solution due to the inferior quality of the initial guess (Scenes c and d). In contrast, as shown in Table II and Fig. 5, our proposed method offers improvement of 53.07%, 76.91% and 76.79% compared with the A*-IPOPT, Hybrid A*-IPOPT and FTHA-IPOPT in Scene a. In Scene b, it has improvement of 82.67% and 82.22% compared with the Hybrid A*-IPOPT and FTHA-IPOPT. In complex Scene c and d, only our method can run efficiently and successfully. In summary, our proposed approach can generate smooth, optimal, and feasible trajectories for parking tasks in both simple and complex scenarios with speed and accuracy.

D. Analysis of the Algorithm Parameters

A series of tests with varying critical parameter settings are conducted to demonstrate algorithm parameters' impact on the proposed planner. As shown in Fig. 8(a), the expansion step size of the first-stage box Δl_1 is increased from 0.1 to 1 and re-simulated; the result demonstrates that the total CPU time decreases as Δl_1 increases, confirming the superiority of our two-stage generation strategy of driving corridor. Fig. 8(b) demonstrates that initial weights $\mathcal{W}_{\text{penalty}}$ have no significant impact on CPU time. Fig. 8(c) and (d) illustrates the optimized trajectory with various sampled finite elements n, and it is discovered that varying n has little influence on parking trajectories. It should be noted that in certain complex environments, n cannot be set too small when the planned trajectory is too long, otherwise the trajectory smoothness may be affected. (e.g., $n = 50$ in Fig. 8(d)).

E. Analysis of the Local State Constraint

This subsection demonstrates the performance of the proposed local state constraint. Based on the map shown in Fig. 5(b), the initial and terminal states are set as $\boldsymbol{x}_{\text{start}}=[-15, -15, 0, 0, 0]$

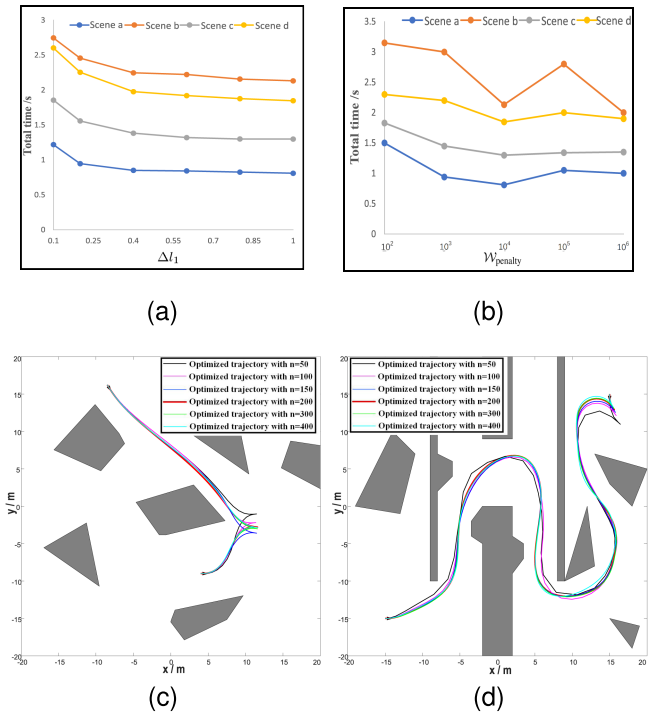


Fig. 8. Performance with different parameter configurations. (a) The total CPU time with varying Δl_1. (b) The total CPU time with varying $\mathcal{W}_{\text{penalty}}$. (c)/(d) The optimized trajectory with varying sampled finite elements n.

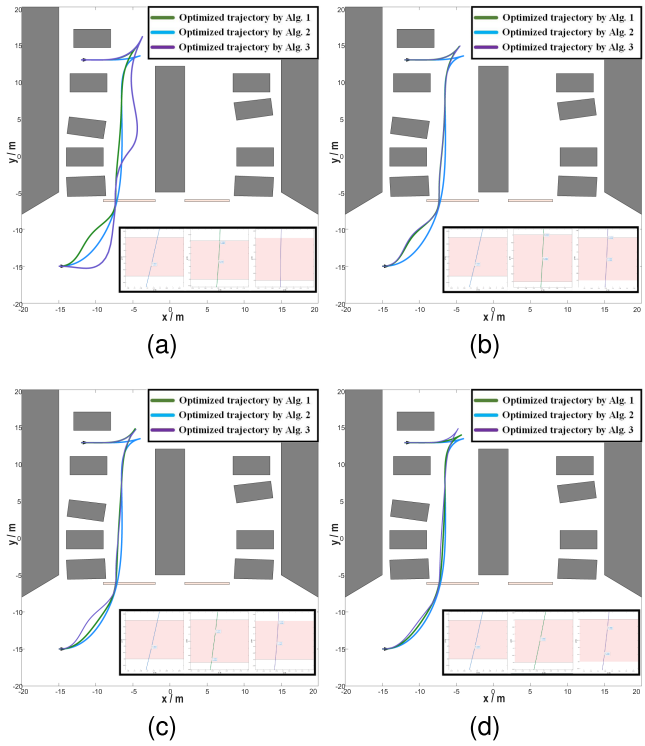


Fig. 9. Optimized trajectory based on different local speed constraints. (a) $0 \leq v_{local}(k) \leq 2$. (b) $0 \leq v_{local}(k) \leq 2.5$. (c) $0 \leq v_{local}(k) \leq 3$. (d) $0 \leq v_{local}(k) \leq 3.5$.

and $\boldsymbol{x}_{\text{final}}=[-12, 12.8, 0, 0, 0]$, respectively. To evaluate the performance of the proposed method, we compare Algorithm 1 (ours) with Algorithm 2 (identical to the proposed Algorithm 1, with the local state constraint (15) removed) and Algorithm 3 (identical to the proposed Algorithm 1, with the local state constraint (15) replaced by (19) in [35]). If the vehicle is to cross the local area occupied by the deceleration strip ($-9 \leq p_x(k) \leq -2$ and $-6.3 \leq p_y(k) \leq -6$ in Fig. 9), the desired vehicle state variable $\mathcal{Z}(k)$ is required that $\mathcal{Z}_a \leq \mathcal{Z}(k) \leq \mathcal{Z}_b$. Taking into account the exact approximation of the original constraint and the limit of the computer's numerical processing [35], the smoothing parameter β in (15) is set to 10 in this article.

This article focuses primarily on the speed of vehicles in local areas. Various velocity constraints ($0 \leq v_{local}(k) \leq 2, 0 \leq v_{local}(k) \leq 2.5$, $0 \leq v_{local}(k) \leq 3$ and $0 \leq v_{local}(k) \leq 3.5$) are imposed in order to evaluate the impact of the approximation formulation. The optimized trajectories and velocity profiles are shown in Figs. 9 and 10, respectively. When moving from the initial position to the final position, the vehicle will cross the deceleration strip to reach the parking space. However, it can be seen from Fig. 10 that Algorithm 2 remains in an accelerated state when passing through the local area, which has a significant impact on the vehicle's comfort. Using the proposed novel approximation formulation (Algorithm 1), the vehicle will decelerate when approaching the local area until it satisfies the local vehicle state requirements ($v_a \leq v_{local}(k) \leq v_b$), and smooth velocity profiles are obtained with these four different velocity constraints. In contrast, Algorithm 3 can achieve a similar performance that satisfies constraint requirements and

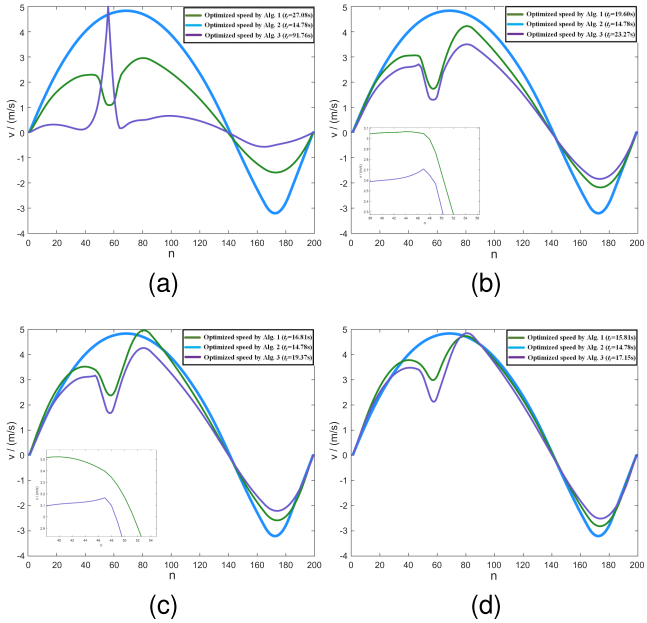


Fig. 10. Optimized velocity based on different local speed constraints. (a) $0 \leq v_{local}(k) \leq 2$. (b) $0 \leq v_{local}(k) \leq 2.5$. (c) $0 \leq v_{local}(k) \leq 3$. (d) $0 \leq v_{local}(k) \leq 3.5$.

smoothness in Fig. 10(d); however, the smoothness of the velocity profile will be affected by the change of the local constraints (Fig. 10(b) and (c)), and may fail to meet the requirements due to solution failure (Fig. 10(a)). Thus, the results demonstrate that Algorithm 3 is highly sensitive to the local constraint, whereas

Algorithm 1 is highly robust. Meanwhile, we can find that even if both Algorithms 1 and 3 can satisfy the constraint requirements and smoothness, we find that our proposed Algorithm 1 can achieve a smaller t_f in the objective function.

In conclusion, it is demonstrated that the proposed novel approximation formulation performs well in dealing with "if-else" structure constraints and has greater adaptability.

V. Conclusion

In this article, a two-stage autonomous valet parking trajectory planning framework is proposed for a complex environment with narrow passages formed by obstacles of different shapes. To efficiently generate a feasible initial guess in above scenes, an enhanced hybrid A* algorithm is designed to solve the search problem in complex environments. The initial guess from the first stage aids the numerical optimization process in the second stage. It is worth noting that the proposed two-stage generation strategy of the driving corridor is used to replace the conventional full-scale collision avoidance constraints to solve optimization problems more efficiently. In the meantime, an novel approximation formulation for handling the local state constraints with an "if-else" structure is introduced. The experimental results demonstrate that the proposed framework has the benefits of high efficiency, accuracy, robustness, optimization, and parameter insensitivity.

This study's proposed framework is limited to parking in complex scenes with a static environment. Future consideration will be given to trajectory planning for autonomous valet parking in a dynamic environment with multiple independent agents [37].

References

[1] S. Luo, X. Li, and Z. Sun, "An optimization-based motion planning method for autonomous driving vehicle," in *Proc. IEEE 3rd Int. Conf. Unmanned Syst.*, 2020, pp. 739–744.

[2] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, "Bench-MR: A motion planning benchmark for wheeled mobile robots," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4536–4543, Jul. 2021.

[3] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous driving on curvy roads without reliance on frenet frame: A cartesian-based trajectory planning method," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15729–15741, Sep. 2022.

[4] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 341–355, Jan. 2021.

[5] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, May 2020.

[6] Y. Guo, D. D. Yao, B. Li, H. Gao, and L. Li, "Down-sized initialization for optimization-based unstructured trajectory planning by only optimizing critical variables," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 709–720, Jan. 2023.

[7] C. Sun, Q. Li, B. Li, and L. Li, "A successive linearization in feasible set algorithm for vehicle motion planning in unstructured and low-speed scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 4, pp. 3724–3736, Apr. 2022.

[8] W. Liu, Z. Li, L. Li, and F.-Y. Wang, "Parking like a human: A direct trajectory planning solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3388–3397, Dec. 2017.

[9] B. Li et al., "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11970–11981, Aug. 2022.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[11] R. Zhou and E. A. Hansen, "Multiple sequence alignment using anytime a *," in *Proc. 18th Nat. Conf. Artif. Intell.*, 2002, pp. 975–977.

[12] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning a *," *Artif. Intell.*, vol. 155, no. 1/2, pp. 93–146, 2004.

[13] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in *Proc. ICAPS*, 2005, pp. 262–271.

[14] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 141–166, 2007.

[15] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 2775–2781.

[16] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on RRT," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 5622–5627.

[17] Y. Dong, E. Camci, and E. Kayacan, "Faster RRT-based nonholonomic path planning in 2D building environments using skeleton-constrained path biasing," *J. Intell. Robotic Syst.*, vol. 89, no. 3, pp. 387–401, 2018.

[18] Y. Lei, Y. Wang, S. Wu, X. Gu, and X. Qin, "A fuzzy logic-based adaptive dynamic window approach for path planning of automated driving mining truck," in *Proc. IEEE Int. Conf. Mechatronics*, 2021, pp. 1–6.

[19] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010 .

[20] P. Zips, M. Böck, and A. Kugi, "Optimisation based path planning for car parking in narrow environments," *Robot. Auton. Syst.*, vol. 79, pp. 1–11, 2016.

[21] K. Bergman and D. Axehill, "Combining homotopy methods and numerical optimal control to solve motion planning problems," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2018, pp. 347–354.

[22] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3263–3274, Nov. 2016.

[23] B. Li and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowl.-Based Syst.*, vol. 86, pp. 11–20, 2015.

[24] B. Li and Z. Shao, "Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots," *Adv. Eng. Softw.*, vol. 87, pp. 30–42, 2015.

[25] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 4327–4332.

[26] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142–153, 2017.

[27] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 613–626, Feb. 2018.

[28] G. Bitar, A. B. Martinsen, A. M. Lekkas, and M. Breivik, "Two-stage optimized trajectory planning for ASVs under polygonal obstacle constraints: Theory and experiments," *IEEE Access*, vol. 8, pp. 199953–199969, 2020.

[29] J. Zhou et al., "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 439–446, Apr. 2021.

[30] B. Li, T. Acarman, Y. Zhang, L. Zhang, C. Yaman, and Q. Kong, "Tractor-trailer vehicle trajectory planning in narrow environments with a progressively constrained optimal control approach," *IEEE Trans. Intell. Veh.*, vol. 5, no. 3, pp. 414–425, Sep. 2020.

[31] B. Li, T. Acarman, X. Peng, Y. Zhang, X. Bian, and Q. Kong, "Maneuver planning for automatic parking with safe travel corridors: A numerical optimal control approach," in *Proc. IEEE Eur. Control Conf.*, 2020, pp. 1993–1998.

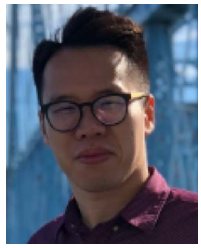
[32] H. Andreasson, J. Saarinen, M. Cirillo, T. Stoyanov, and A. J. Lilienthal, "Fast, continuous state path smoothing to improve navigation accuracy," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 662–669.

[33] C. Chen, M. Rickert, and A. Knoll, "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2015, pp. 1148–1153.

[34] W. Sheng, B. Li, and X. Zhong, "Autonomous parking trajectory planning with tiny passages: A combination of multistage hybrid a-star algorithm and numerical optimal control," *IEEE Access*, vol. 9, pp. 102801–102810, 2021.
[35] Y. Guo, D. Yao, B. Li, Z. He, H. Gao, and L. Li, "Trajectory planning for an autonomous vehicle in spatially constrained environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 18326–18336, Oct. 2022.
[36] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.
[37] J. Leu, Y. Wang, M. Tomizuka, and S. Di Cairano, "Autonomous vehicle parking in dynamic environments: An integrated system with prediction and motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 10890–10897.

Jing Lian (Member, IEEE) received the Ph.D. degree in communication and information system from Jilin University, Jilin, China, in 2008. She is currently an Associate Professor and the Deputy Director of the Automotive Electronic Institute, Dalian University of Technology, Dalian, China, and a Judicial Expert in the vehicle performance. She is the Leader of more than 20 research projects. She is the author of more than 70 publications. Her main research interests include vision based environmental perception, automotive electronics, trajectory planning, and control of intelligent vehicle.

Weiwei Ren received the B.S. degree from the Harbin Institute of Technology, Weihai, China, in 2019. He is currently working toward the Ph.D. degree with the Dalian University of Technology, Dalian, China. His research interests include decision-making and trajectory planning of autonomous vehicles.



Dongfang Yang received the B.E. degree in microelectronics from Sun Yat-sen University, Guangzhou, China, in 2014, and the M.S. and Ph.D. degrees in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2019 and 2020, respectively. He is currently a Senior Algorithm Engineer with Chongqing Chang'an Automobile Company, Ltd., and a Postdoc Researcher with Chongqing University, Chongqing, China. His research interests include data analysis, machine learning, deep learning, and control, with applications in behavior prediction, decision-making, and motion planning of autonomous systems.

Linhui Li (Member, IEEE) received the Ph.D. degree in vehicle operation engineering from Jilin University, Jilin, China, in 2008. He is currently an Associate Professor with the Dalian University of Technology, Dalian, China. From 2017 to 2018, he was a Visiting Scholar with The Ohio State University, Columbus, OH, USA. He is the author of more than 40 publications. His main research interests include intelligent vehicle trajectory planning, vision based environmental perception of intelligent vehicle, and navigation control.

Fengning Yu received the B.S. degree from the Liaoning University of Technology, Jinzhou, China, in 2017. He is currently working toward the Ph.D. degree with the Dalian University of Technology, Dalian, China. His main research interests include autonomous vehicle environmental sensing, computer vision, and trajectory planning.