Optimization-Based Trajectory Planning for Autonomous Parking With Irregularly Placed Obstacles: A Lightweight Iterative Framework

Bai Li, *Member, IEEE*, Tankut Acarman, *Member, IEEE*, Youmin Zhang, *Senior Member, IEEE*, Yakun Ouyang, *Student Member, IEEE*, Cagdas Yaman, Qi Kong, *Member, IEEE*, Xiang Zhong, and Xiaoyan Peng

***Abstract*— This paper is focused on planning fast, accurate, and optimal trajectories for autonomous parking. Nominally, this task should be described as an optimal control problem (OCP), wherein the collision-avoidance constraints guarantee travel safety and the kinematic constraints guarantee tracking accuracy. The dimension of the nominal OCP is high because it requires the vehicle to avoid collision with each obstacle at every moment throughout the entire parking process. With a coarse trajectory guiding a homotopic route, the intractably scaled collision-avoidance constraints are replaced by within-corridor constraints, whose scale is small and independent from the environment complexity. Constructing such a corridor sacrifices partial free spaces, which may cause loss of optimality or even feasibility. To address this issue, our proposed method reconstructs the corridor in an iterative framework, where a lightweight OCP with only box constraints is quickly solved in each iteration. The proposed planner, together with several prevalent optimization-based planners are tested under 115 simulation cases w.r.t. the success rate and computational time. Real-world indoor experiments are conducted as well.**

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

I. Introduction

A. Background

AUTONOMOUS driving systems are making a promising change to the urban transportation modes [1]. As a critical module in an autonomous vehicle [2], [3], trajectory planning identifies a spatiotemporal curve that is free from collisions, easily tracked by the controller, and comfortable for the passengers. Trajectory planners have been widely developed for on-road autonomous driving [4]–[6]. Compared with on-road cruising, parking in an unstructured scenario is more challenging [7]–[9] because 1) there is no longer a navigational reference line, 2) a trajectory typically contains cusps, thus it is no longer a smooth curve, and 3) obstacles are less regularized than those on a structured road. Few of the prevalent on-road trajectory planners can be used to handle a generic parking planning scheme because of these factors. This work is focused on the autonomous parking trajectory planning problems in complex scenarios with irregularly placed static obstacles.

B. Related Works

The existing trajectory planners suitable for autonomous parking primarily include the sampling-and-search-based and optimization-based methods. A sampling-and-search-based approach discretizes the continuous state space into a graph of nodes and searches in the graph for a valid link from the starting node to the goal node. This category has two branches, that is, to sample the state space or control space. Typical methods that sample the state space primarily include the state lattice approach [10], and the rapidly-exploring random tree series [11], [12]. Meanwhile, the typical control-space samplers include the hybrid A* algorithm [13], and the dynamic window approach [14]. As the second category, optimization-based methods describe a trajectory planning task as an optimal control problem (OCP) and discretize it into a nonlinear programming (NLP) problem [15], [16]. Sampling-and-search- based planners are good at choosing the homotopy class globally (e.g., left or right around each of the obstacles) while optimization-based planners are superior in finding a local optimum while keeping the homotopy class not altered. Sampling-and-search-based and optimization-based planners perform better when they are combined. The rest of this subsection elaborates on the state-of-the-art optimization-based trajectory planners for autonomous parking.

An optimization-based trajectory planner is commonly used to provide optimal trajectories precisely. To the best of the authors' knowledge, Kondak and Hommel [17] were the first to describe a parking trajectory planning problem as an OCP and then numerically solve it. Using a triangle-area criterion

Manuscript received 23 June 2020; revised 22 January 2021, 8 July 2021, and 2 August 2021; accepted 27 August 2021. Date of publication 8 September 2021; date of current version 9 August 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62103139 and Grant 61873047, in part by the Fundamental Research Funds for Central Universities of China under Grant 531118010509, in part by the Galatasaray University Scientific Research Support under Grant 19.401.005, and in part by the Natural Sciences and Engineering Research Council of Canada under Grant RGPIN-2017-06680. The Associate Editor for this article was S. Rathinam. *(Corresponding author: Xiang Zhong.)*

Bai Li, Yakun Ouyang, Xiang Zhong, and Xiaoyan Peng are with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: libai@hnu.edu.cn; yakun@hnu.edu.cn; zx5587@126.com; xiaoyan_p@126.com).

Tankut Acarman and Cagdas Yaman are with the Department of Computer Engineering, Galatasaray University, Istanbul 34349, Turkey (e-mail: tacarman@gsu.edu.tr; truckercy@hotmail.com).

Youmin Zhang is with the Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: ymzhang@encs.concordia.ca).

Qi Kong is with the JDX Research and Development Center, JD.com American Technologies Corporation, Mountain View, CA 94043 USA (e-mail: spiritkong@gmail.com).

This article has supplementary downloadable material available at https://doi.org/10.1109/TITS.2021.3109011, provided by the authors.

Digital Object Identifier 10.1109/TITS.2021.3109011

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

to describe the vehicle-to-obstacle collision-avoidance constraints analytically, Li and Shao [18] proposed a unified OCP model for parking cases with arbitrarily placed obstacles. However, such collision-avoidance constraints are nominally non-differentiable and non-convex. Shi *et al.* [19] formulated a nested optimization model to simplify the nominal collision-avoidance constraints. Zhang *et al.* [20] proposed an optimization-based collision avoidance (OBCA) method, wherein signed distance functions are used to improve the differentiability of the nominal collision-avoidance constraints. Bergman and Axehill [21] proposed a Sequential Homotopy Quadratic Programming (SHQP) strategy, which shrinks the obstacles to ease the collision-avoidance constraints and then expands them incrementally towards their nominal sizes. By solving a series of easier OCPs prior to handling the nominal one, the entire difficulties are dispersed evenly among the easier OCPs. A similar idea was proposed by Li *et al.* [22], wherein an arbitrary spatiotemporal restriction is first imposed to simplify the OCP formulation, and then gradually relaxed until a near-optimal parking trajectory is found.

A near-optimal or even near-feasible initial guess can largely facilitate the numerical solution process of an OCP. One may generate a coarse trajectory and then warm-start an OCP solution process with that initial guess. A* [23], hybrid A* [13], and metaheuristic algorithms [24] were used to provide an initial guess. Since the predominant NLP solvers only find local optima, an initial guess determines the homotopy class of the optimized trajectory finally [25]. The same-homotopy-class property makes an initial guess close to the final optimum, thus it is natural to consider paving a local corridor along the initial guess so that the vehicle is completely separated from the surrounding obstacles [26]. Circles [27], rectangles [28], and convex polygons [29] were considered to construct a corridor, whereby the intractably scaled collision-avoidance constraints in the nominal OCP are converted into a small scale of within-corridor constraints. However, these methods either ignore the vehicle shape when paving the corridor [27], [29] or require exhausting offline preparations [28]. Li et al [30] proposed a spiral growth strategy to construct the corridors along a reference trajectory derived by the hybrid A* algorithm online, thereby quickly finding a near-optimal parking trajectory. However, this method sometimes fails for the following reasons. First, the adopted hybrid A* method is theoretically incomplete, thus the corridors are unavailable if the hybrid A* algorithm becomes inefficient. Second, the reference trajectory derived by the hybrid A* method may not be optimal or even kinematically feasible, thus influencing the quality of the constructed corridors.

C. Contributions

This work aims to achieve fast, accurate, and high-quality parking trajectory planning performance. To ensure that the planning results are optimal and precise, we need to formulate an OCP and numerically solve it. To make the numerical solution process fast, we need to 1) identify a good initial guess via a sampling-and-search-based method quickly; and 2) replace the intractably scaled collision-avoidance constraints with within-corridor conditions. As previously mentioned, constructing corridors has the underlying risks of rendering the OCP infeasible or the derived solution far from being optimal. As a typical sampling-and-search-based algorithm for autonomous parking, the conventional hybrid A* algorithm is incomplete, thus sometimes cannot find a solution within a limited time. The purpose of this work is to address the aforementioned issues in planning fast, accurate, and high-quality trajectories for autonomous parking.

The contributions of this paper are two-fold. First, a fault-tolerant variant of the hybrid A* algorithm is proposed, which ensures that a reference trajectory is available quickly even if the conventional hybrid A* algorithm becomes inefficient. Second, an iterative framework that incorporates a corridor construction procedure and a lightweight NLP solution procedure is proposed to ensure that the trajectory optimization quality is not affected when the initial guess is poor, or when the free spaces are left out by the constructed corridors.

D. Organization

The rest of this paper is structured as follows. Section II presents the problem statement in the form of a nominal OCP. As a foundation, Section III introduces how to convert the nominal collision-avoidance constraints of the OCP into within-corridor constraints. A lightweight iterative framework is proposed in Section IV for providing trajectory planning results. Experimental setups and results are elaborated in Section V. Finally, some conclusions are drawn in Section VI.

II. Problem Formulation

In this section, a generic autonomous parking trajectory planning scheme is formulated as an OCP nominally.

A. Overall Formulation

Let us denote the vehicle state profile as $\mathbf{z} \in \mathbb{R}^{n_z}$, the control profile as $\mathbf{u} \in \mathbb{R}^{n_u}$, the workspace as Υ, the obstacle space as $\Upsilon_{\text{OBS}} \subset \Upsilon$, and the free space as $\Upsilon_{\text{FREE}} = \Upsilon \backslash \Upsilon_{\text{OBS}}$. The trajectory planning scheme is written as

$$\begin{aligned} \min_{\mathbf{z}(t),\mathbf{u}(t),T} \; & J\left(\mathbf{z}(t),\mathbf{u}(t)\right), \\ \text{s.t.,} \; & \dot{\mathbf{z}}(t) = f\left(\mathbf{z}(t),\mathbf{u}(t)\right), \\ & \mathbf{z} \le \mathbf{z}(t) \le \bar{\mathbf{z}}, \quad \mathbf{u} \le \mathbf{u}(t) \le \bar{\mathbf{u}}, \; t \in [0,T]; \\ & \mathbf{z}(0) = \mathbf{z}_{\text{init}}, \quad \mathbf{u}(0) = \mathbf{u}_{\text{init}}; \\ & g_{\text{end}}\left(\mathbf{z}(T), \quad \mathbf{u}(T)\right) \le 0; \\ & fp(\mathbf{z}(t)) \subset \Upsilon_{\text{FREE}}, \quad t \in [0,T]. \end{aligned} \tag{1}$$

Herein, T denotes the parking process duration in seconds (not known *a priori*), and J is the cost function to be minimized. We use the common shorthand $\dot{\mathbf{z}}$ to denote the derivative w.r.t. time, i.e., $\dot{\mathbf{z}} = \partial \mathbf{z}/\partial t$ [31], [42]. Function f describes the vehicle kinematics. $[\mathbf{z}, \bar{\mathbf{z}}]$ and $[\mathbf{u}, \bar{\mathbf{u}}]$ denote the allowable intervals where $\mathbf{z}(t)$ and $\mathbf{u}(t)$ stay. $\mathbf{z}_{\text{init}}$ and $\mathbf{u}_{\text{init}}$ denote the initial values of $\mathbf{z}(t)$ and $\mathbf{u}(t)$, respectively. The inequality $g_{\text{end}} \le 0$ models the implicit end-point conditions at $t = T$. $fp(\cdot)$:$\mathbb{R}^{n_z} \to \mathbb{R}^2$ is a mapping from vehicle

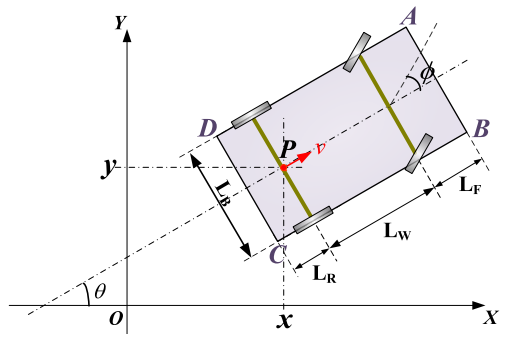


Fig. 1. Schematics on vehicle kinematics and geometric sizes.

state to its footprints, thus $fp(\mathbf{z}(t)) \subset \Upsilon_{\text{FREE}}, \forall t$ represents the collision- avoidance constraints. In the remainder of this section, we elaborate on the details behind the aforementioned abstract symbols.

B. Vehicle Kinematic Constraints

This subsection formulates the differential equations $\dot{\mathbf{z}}(t) = f(\mathbf{z}(t), \mathbf{u}(t))$ in (1). The well-known bicycle model is used to describe the vehicle kinematics during the low-speed parking process [18]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x(t) \\ y(t) \\ v(t) \\ \phi(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} v(t)\cdot\cos\theta(t) \\ v(t)\cdot\sin\theta(t) \\ a(t) \\ \omega(t) \\ v(t)\cdot\tan\phi(t)/\mathrm{L_W} \end{bmatrix}, \quad t \in [0, T]. \tag{2}$$

Herein, (x, y) is the rear-wheel axle mid-point (point P in Fig. 1), ϕ is the steering angle, ω is the angular velocity, v is the velocity of P, a stands for the acceleration, θ is the orientation angle, and $\mathrm{L_W}$ denotes the wheelbase. According to (1), the state profiles $\mathbf{z}$ refer to the variables that are differentiated. Thus $\mathbf{z}(t) = [x(t), y(t), \theta(t), v(t), \phi(t)]$ and $\mathbf{u}(t) = [a(t), \omega(t)]$.

Besides $\mathrm{L_W}$, other geometric parameters such as $\mathrm{L_F}$ (front overhang length), $\mathrm{L_R}$ (rear overhang length), and $\mathrm{L_B}$ (width) are also depicted in Fig. 1.

The aforementioned state/control variables have allowable intervals, which reflect the physical or mechanical limitations related to vehicle kinematics:

$$\begin{bmatrix} \mathrm{a_{min}} \\ \mathrm{v_{min}} \\ -\Omega_{\max} \\ -\Phi_{\max} \end{bmatrix} \le \begin{bmatrix} a(t) \\ v(t) \\ \omega(t) \\ \phi(t) \end{bmatrix} \le \begin{bmatrix} \mathrm{a_{max}} \\ \mathrm{v_{max}} \\ \Omega_{\max} \\ \Phi_{\max} \end{bmatrix}, \quad t \in [0, T]. \tag{3}$$

C. Boundary Constraints

Boundary constraints $\mathbf{z}(0) = \mathbf{z}_{\text{init}}$, $\mathbf{u}(0) = \mathbf{u}_{\text{init}}$, and $g_{\text{end}} \le 0$ in (1) are deployed to specify the vehicle's configurations at $t = 0$ and T. At $t = 0$, we have

$$[x(0), y(0), \theta(0), v(0), \phi(0), a(0), \omega(0)] = [\mathrm{x_0}, \mathrm{y_0}, \theta_0, \mathrm{v_0}, \phi_0, \mathrm{a_0}, \mathrm{w_0}], \tag{4}$$

Nonetheless, at the endpoint $t = T$, simply imposing $\mathbf{z}(T) = \mathbf{z}_{\text{end}}$ may cause problems because of $\theta(T)$. Concretely, $\theta(t)$ is differentiable, thus it is continuous w.r.t. t. Since

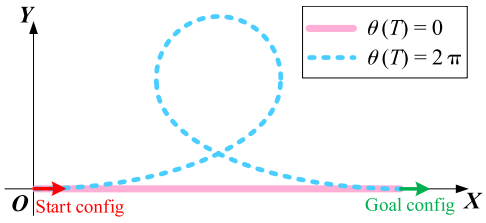


Fig. 2. An example reflecting the difference between setting $\theta(T)$ to 0 and 2π.

$\theta(t)$ cannot jump, directly setting $\theta(T)$ to a point would fix the phase difference between $\theta(0)$ and $\theta(T)$, which might misleadingly restrict the behavior of the vehicle. For example, the trajectories of a vehicle that travels from $\theta(0) = 0$ to $\theta(T) = 0$ or $\theta(T) = 2\pi$ are plotted in Fig. 2, respectively. Typically in an on-road local trajectory planning scheme, the phase difference between $\theta(0)$ and $\theta(T)$ is not large, thus setting $\theta(T)$ to a point never excites challenges. But free-space parking allows backward maneuvers, thus a winding trajectory may be necessary to avoid collisions. Due to this reason, we use the following equalities to describe the end-point constraints:

$$[x(T), y(T), v(T), \phi(T), a(T), \omega(T)] = [\mathrm{x_F}, \mathrm{y_F}, \mathrm{v_F}, \phi_{\mathrm{F}}, \mathrm{a_F}, \mathrm{w_F}], \tag{5a}$$

and

$$\begin{aligned} \sin\theta(T) &= \sin(\theta_{\mathrm{F}}), \\ \cos\theta(T) &= \cos(\theta_{\mathrm{F}}). \end{aligned} \tag{5b}$$

D. Collision-Avoidance Constraints

This subsection presents the collision-avoidance constraints $fp(\mathbf{z}(t)) \subset \Upsilon_{\text{FREE}}$ in (1). Suppose that the ego vehicle moves in a static environment with $\mathrm{N_{OBS}}$ convex polygonal obstacles. Collisions between the ego vehicle and each obstacle should be avoided at every instance during $[0, T]$. Suppose that the ego vehicle does not collide with any of the obstacles at the initial moment $t = 0$, collisions will not occur during $t \in (0, T]$ if 1) every vertex of the rectangular ego vehicle stays out of each polygonal obstacle, and 2) every vertex of each polygonal obstacle stays out of the rectangular ego vehicle. It deserves to emphasize that the aforementioned conditions are true only when the polygonal obstacles are convex. Concave polygonal obstacles need to be decomposed into convex ones. The requirement to keep one point $\mathrm{Q} = (x_{\mathrm{q}}, y_{\mathrm{q}})$ out of a polygon $\mathrm{W_1W_2 \ldots W_m}$ can be modeled analytically via a triangle-area criterion [32]:

$$S_{\Delta \mathrm{QW_mW_1}} + \sum_{l=1}^{\mathrm{m}-1} S_{\Delta \mathrm{QW}_l\mathrm{W}_{l+1}} > S_{\square \mathrm{W_1W_2 \ldots W_m}}, \tag{6}$$

where S_Δ denotes a triangle area, and $S_\square$ denotes the area of a polygon. Each S_Δ is computed based on the vertex coordinates. Concretely, suppose that the coordinate of W_l is

$(\mathrm{x}_{\mathrm{W}l}, \mathrm{y}_{\mathrm{W}l})$, then

$$S_{\Delta \mathrm{QW}_l \mathrm{W}_{(l+1)}} = \frac{1}{2} \cdot \left| x_{\mathrm{q}} \mathrm{y}_{\mathrm{W}l} + \mathrm{x}_{\mathrm{W}l} \mathrm{y}_{\mathrm{W}(l+1)} + \mathrm{x}_{\mathrm{W}(l+1)} y_{\mathrm{q}} - x_{\mathrm{q}} \mathrm{y}_{\mathrm{W}(l+1)} - \mathrm{x}_{\mathrm{W}l} y_{\mathrm{q}} - \mathrm{x}_{\mathrm{W}(l+1)} \mathrm{y}_{\mathrm{W}l} \right|. \quad (7)$$

As a constant value, $S_{\triangleright \mathrm{W}_1 \mathrm{W}_2 \ldots \mathrm{W}_\mathrm{m}}$ is calculated by summing up multiple triangle areas.

Suppose that the four vertexes of the ego vehicle at t are denoted as $A(t), B(t), C(t),$ and $D(t)$, and that the jth obstacle has NP_j vertexes $\{\mathrm{V}_{jk} \,|k = 1, \ldots, \mathrm{NP}_j\}$, then the conditions 1) and 2) are summarized into the following inequalities:

$$S_{\Delta Q \mathrm{V}_{j\mathrm{NP}_j} \mathrm{V}_{j1}} + \sum_{k=1}^{\mathrm{NP}_j - 1} S_{\Delta Q \mathrm{V}_{jk} \mathrm{V}_{j(k+1)}} > S_{\triangleright \mathrm{V}_{j1} \mathrm{V}_{j2} \ldots \mathrm{V}_{j\mathrm{NP}_j}}, \times \forall Q \in \{A(t), B(t), C(t), D(t)\}, \quad (8)$$

and

$$S_{\Delta QA(t)B(t)} + S_{\Delta QB(t)C(t)} + S_{\Delta QC(t)D(t)} + S_{\Delta QD(t)A(t)} > S_{\triangleright A(t)B(t)C(t)D(t)}, \times \forall Q \in \left\{\mathrm{V}_{j1}, \mathrm{V}_{j2}, \ldots, \mathrm{V}_{j\mathrm{NP}_j}\right\}. \quad (9)$$

Applying (8) and (9) to $\forall t \in [0, T]$ and $j = 1, \ldots, \mathrm{N_{OBS}}$ yields the complete collision-avoidance constraints.

This work assumes that all of the obstacles are static because predicting the future maneuver of a moving obstacle in a parking scenario is typically not as easy as it is on a structured road. Also, some extra computational time is needed before the vehicle can really react to the moving obstacles. In parking an autonomous vehicle, the primary challenge is not the capability to consider moving obstacles, but to handle the irregularly placed obstacles in a cluttered environment regardless of they are static or not.

E. Cost Function

In this work, a time-energy cost function is defined as follows:

$$J = \mathrm{w} \cdot \int_{\tau=0}^{T} \left(a^2(\tau) + v^2(\tau) \cdot \omega^2(\tau) \right) \cdot \mathrm{d}\tau + T, \quad (10)$$

where $\mathrm{w} > 0$ is a weighting parameter. The first term models the passenger comfort according to the ISO 2631-1 standard mentioned in [33], the second term is deployed to show our preference to finish the parking process early.

As a summary of the whole section, a generic autonomous parking trajectory planning scheme is described as the following OCP:

$$\begin{aligned} &\min_{\mathbf{z}(t), \mathbf{u}(t), T} \quad (10), \\ &\text{s.t. Kinematic constraints (2) and (3);} \\ &\quad \text{Boundary conditions (4) and (5);} \\ &\quad \text{Collision-avoidance constraints (8) and (9).} \end{aligned} \quad (11)$$

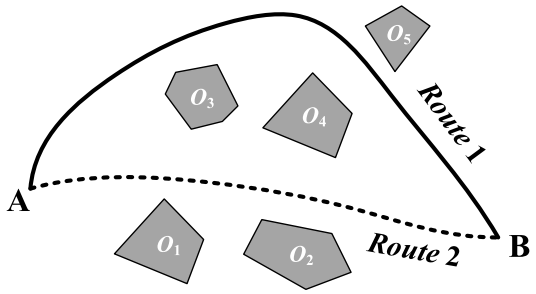


Fig. 3. A simple case illustrating partial collision-avoidance constraints in the nominal OCP formulation are redundant.

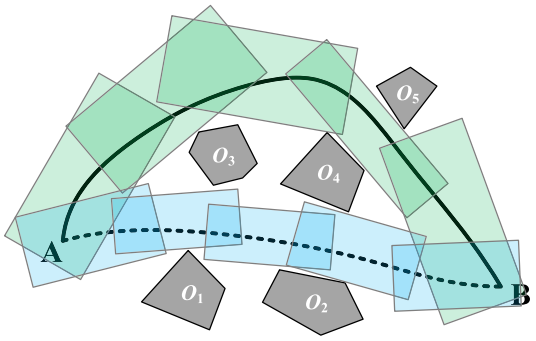


Fig. 4. Schematics of the constructed corridors. A corridor is paved along each route and each corridor consists of five local boxes. Note that the scale of the within-corridor constrains is irrelevant to the number of obstacles.

III. Corridor Construction

A. Motivations

The nominal OCP (11) formulated in the preceding section is intractable because it requires the ego vehicle to avoid collision with each obstacle at every time instant. However, the ego vehicle does not have the chance to collide with *each* obstacle at *every* moment. Regarding the task to travel from A to B in Fig. 3, if the ego vehicle travels via route 1, then it has slim chances to collide with obstacles O_1 or O_2 at the bottom; if the vehicle chooses route 2 instead, it may not collide with O_5. Therefore, simultaneously avoiding collisions with O_1, O_2, and O_5 renders redundant constraints. If a guiding route is given *a priori*, the scale of the collision-avoidance constraints can be reduced. To consider further, if a corridor is constructed along a selected route to separate the ego vehicle from the surrounding obstacles, then the scale of the environment-related constraints becomes fully independent from the number of obstacles, i.e., being irrelevant to the complexity of the environment (Fig. 4).

In the remainder of this section, we first identify a guiding route, then construct the corridors, finally formulate the within-corridor constraints.

B. Generation of a Guiding Route

A guiding route like the ones presented in Fig. 4 should be identified before we construct the corridors. This subsection introduces how to generate such a guiding route. Searching in the $x - y - \theta$ 3-dim mesh grids, the hybrid A* algorithm [13]

generates a route that connects the initial and goal configurations.

Since the hybrid A* algorithm is not complete, it may fail when the environment is complex. We propose a fault-tolerant hybrid A* (FTHA) algorithm, which guarantees to have an output even if the conventional hybrid A* search fails. In FTHA, a best-so-far node is maintained to record the explored node with the smallest cost-to-go value during the iterations. Recall that a hybrid A* search process is regarded as failed when the maximum iteration number is reached or the openlist becomes empty. When a hybrid A* search process fails, one can still get a path from the initial node to the best-so-far node. At the same time, another path from the best-so-far node to the goal node can be identified via a 2-dim A* search. Connecting the two paths renders a guiding route from the initial point to the goal point, although the end-point orientation angle may not satisfy the desired constraint (5b). The identified guiding route determines from which side the ego vehicle bypasses each of the $\mathrm{N_{OBS}}$ static obstacles.

When the guiding route is derived, we attach a minimum-time velocity profile to form a coarse trajectory. This is achieved by solving a one-dimensional OCP via Pontryagin's Maximum Principle (PMP). The derived coarse trajectory is resampled evenly in time as $(\mathrm{N_{FE}}+1)$ elements $\chi_P = \{\sigma_0, \sigma_1, \ldots, \sigma_{N_{FE}}\}$, wherein each element σ_i is a vector containing the vehicle location, orientation angle, and time stamp, i.e., $\sigma_i = [x(t_i), y(t_i), \theta(t_i), t_i]$. The time-stamp in the last element $t_{N_{FE}}$ reflects the initially guessed T, which is specified by the aforementioned PMP.

Note that a coarse trajectory derived by our FTHA algorithm may not satisfy the vehicle kinematic constraints, but as we will introduce later, χ_P is used to construct the corridors *at the very beginning*. This means that the underlying low quality of χ_P would not influence the entire trajectory optimization performance *provided that* the selected homotopy class is unpromising.

C. Construction of Corridors

This subsection introduces the construction of corridors along the coarse trajectory such that the ego vehicle can safely avoid all of the obstacles if it stays in the corridors. When a corridor is available, requiring a mass point to stay in the corridor is easier than requiring a rectangular vehicle. Thus the well-known safe flight corridor (SFC) based planners in the community of unnamed aerial vehicle (UAV) (e.g. [34], [35]) are not directly applicable. In our study, the SFC concept is extended such that a rectangular vehicle body can be dealt with.

Two discs are used to evenly cover the vehicle body (Fig. 5a). The disc centers, namely $P_\mathrm{f} = (x_\mathrm{f}, y_\mathrm{f})$ and $P_\mathrm{r} = (x_\mathrm{r}, y_\mathrm{r})$, are quartile points along the vehicle's longitudinal axle:

$$\begin{aligned} x_\mathrm{f}(t) &= x(t) + \frac{1}{4}(3\mathrm{L_W} + 3\mathrm{L_F} - \mathrm{L_R}) \cdot \cos\theta(t), \\ y_\mathrm{f}(t) &= y(t) + \frac{1}{4}(3\mathrm{L_W} + 3\mathrm{L_F} - \mathrm{L_R}) \cdot \sin\theta(t), \\ x_\mathrm{r}(t) &= x(t) + \frac{1}{4}(\mathrm{L_W} + \mathrm{L_F} - 3\mathrm{L_R}) \cdot \cos\theta(t), \end{aligned}$$

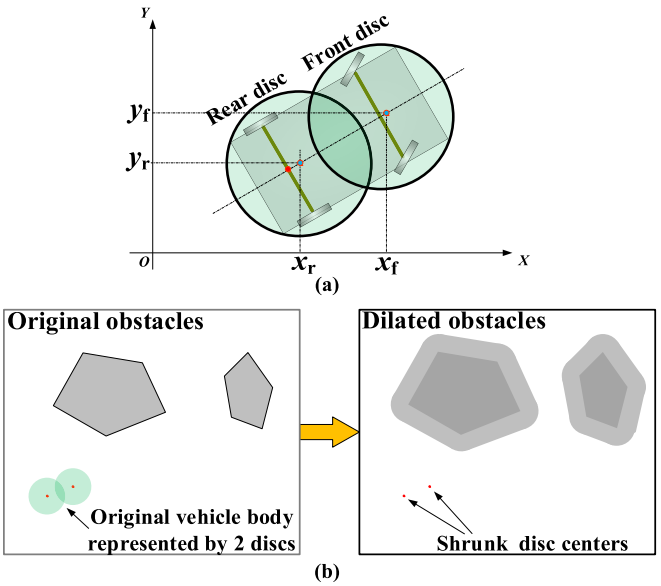


Fig. 5. Schematics of vehicle shape shrinking and obstacle dilatation: (a) presenting the vehicle shape with two discs; (b) shrinking the vehicle shape and dilating the environmental obstacles.

$$y_\mathrm{r}(t) = y(t) + \frac{1}{4}(\mathrm{L_W} + \mathrm{L_F} - 3\mathrm{L_R}) \cdot \sin\theta(t). \tag{12}$$

$\mathrm{R_D}$, the radius of either disc, is determined by

$$\mathrm{R_D} = \frac{1}{2}\sqrt{(\frac{\mathrm{L_R} + \mathrm{L_W} + \mathrm{L_F}}{2})^2 + (\mathrm{L_B})^2}. \tag{13}$$

The condition that each disc does not overlap with the obstacles is the same as that each disc center keeps a distance of $\mathrm{R_D}$ from the obstacles at least. Therefore, an equivalent conversion can be made by simultaneously shrinking the two discs to their centers and dilating the obstacles by $\mathrm{R_D}$ (see Fig. 5b). The new map with the dilated obstacles is denoted as a *dilated map*.

The next step is to construct two corridors in the dilated map for P_f and P_r, respectively. Note that the corridors we build for the disc centers are based on the dilated map, thus keeping P_f and P_r in their own series of corridors ensures that the rectangular vehicle does not collide with the obstacles in the original environment.

Let us focus on the corridor construction process of P_f first. With the coarse trajectory χ_P at hand, we compute the coarse trajectory of point P_f via (12). The resultant trajectory is presented in the form of $(\mathrm{N_{FE}}+1)$ elements $\chi_{P_\mathrm{f}} = \{\sigma_0^\mathrm{F}, \sigma_1^\mathrm{F}, \ldots, \sigma_{N_{FE}}^\mathrm{F}\}$. We need to pave a corridor which 1) fully covers χ_{P_f}, and 2) does not overlap with the dilated obstacles in the dilated map. We require that the corridor consists of multiple regularly placed local boxes like the ones shown in Fig. 6a. We use regularly placed local boxes because they make the within-corridor constraints box constraints (we will introduce this idea in the next subsection). Except for the first element, each of the rest $\mathrm{N_{FE}}$ elements in χ_{P_f} represents a point, which is called a *waypoint*. Each waypoint should be covered by one local box, which means a corridor consists of $\mathrm{N_{FE}}$ local boxes. Without losing generality, we focus on how to identify the kth local box.

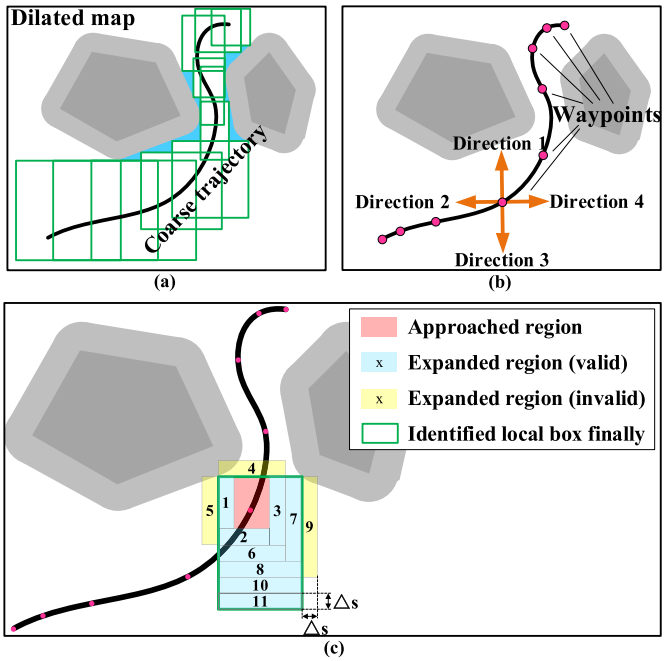


Fig. 6. Principle to identify a corridor: (a) a corridor consisting of local boxes; (b) expansion trial direction order and sampled waypoints along the coarse trajectory; (c) expansion trials conducted in a sequence.

The kth local box is responsible for covering the element indexed k in χ_{P_f}. The kth local box is initially constituted by a zero-width-zero-length rectangle and is expanded in each edge in turn until an overlap with the dilated map is detected. The expansion order is defined in Fig. 6b. In Fig. 6c, each expansion trial pushes the involved edge of the currently approved region outwards by a constant size Δs. We check, at each time, whether the expanded region (see each small block labeled with a unique number 1,2,3,…,11 in Fig. 6c) causes collisions in the dilated map [36]. If no collision occurs, the expanded region is merged into the currently approved region; otherwise, it is rejected and no further expansions are conducted in that direction. In each direction, a maximum expansion length $\mathrm{L_{limit}}$ is introduced to avoid excessive growth. The principle to identify a local box over a given waypoint $(\mathrm{x_c}, \mathrm{y_c})$ is summarized by the pseudo-codes in Alg. 1 as follows:

The output of GenerateLocalBox() is a vector containing four elements, which records the ranging intervals of the identified local box in the x and y axes, respectively. In line 7, the function $trial\_area \leftarrow \mathrm{ExpandBox}(box, dir)$ is used to identify an additional region $trial\_area$ by pushing the edge of a regularly placed box box in the direction of dir by a certain step Δs. The function IsBoxValid($trial\_area$) is deployed to check if the aforementioned region $trial\_area$ overlaps with the dilated obstacles in the dilated map.

Applying Alg. 1 to all of the waypoints yields two corridors for P_f and P_r, respectively:

The output of GenerateCorridors() is a vector containing $8\mathrm{N_{FE}}$ elements, all of which determine the two corridors. In line 1 of Alg. 2, the function GenerateReferenceTrajectory(χ_P) calculates two reference trajectories according to the coarse trajectory χ_P we identify in Section III.B.

Algorithm 1 Local Box Generation

Function GenerateLocalBox($\mathrm{x_c}, \mathrm{y_c}$)

1. $\Omega_{\mathrm{ID}} \leftarrow \{1, 2, 3, 4\}$
2. $\Omega_{\mathrm{DIRECTION}} \leftarrow \{\pi/2, \pi, 3\pi/2, 0\}$;
3. $\Omega_{\mathrm{LENGTH}} \leftarrow \{0, 0, 0, 0\}$;
4. $\Upsilon_{\mathrm{APPROVED}} \leftarrow (\mathrm{x_c}, \mathrm{y_c})$;
5. **while** $\Omega_{\mathrm{ID}} \neq \varnothing$ **do**
6. **for each** $i \in \Omega_{\mathrm{ID}}$ **do**
7. $\Upsilon_{\mathrm{TRIAL}} \leftarrow \mathrm{ExpandBox}(\Upsilon_{\mathrm{APPROVED}}, \Omega_{\mathrm{DIRECTION}}[i])$;
8. **if** IsBoxValid($\Upsilon_{\mathrm{TRIAL}}$) **then**
9. $\Omega_{\mathrm{LENGTH}}[i] \leftarrow \Omega_{\mathrm{LENGTH}}[i] + \Delta\mathrm{s}$;
10. $\Upsilon_{\mathrm{APPROVED}} \leftarrow \Upsilon_{\mathrm{TRIAL}} \cup \Upsilon_{\mathrm{APPROVED}}$;
11. **if** $\Omega_{\mathrm{LENGTH}}[i] \geq \mathrm{L_{limit}}$ **then**
12. $\Omega_{\mathrm{ID}} \leftarrow \Omega_{\mathrm{ID}} \backslash i$;
13. **end if**
14. **else**
15. $\Omega_{\mathrm{ID}} \leftarrow \Omega_{\mathrm{ID}} \backslash i$;
16. **end if**
17. **end for**
18. **end while**
19. $\mathrm{x_{min}} \leftarrow \mathrm{x_c} - \Omega_{\mathrm{LENGTH}}[3]$, $\mathrm{x_{max}} \leftarrow \mathrm{x_c} + \Omega_{\mathrm{LENGTH}}[1]$, $\mathrm{y_{min}} \leftarrow \mathrm{y_c} - \Omega_{\mathrm{LENGTH}}[2]$, $\mathrm{y_{max}} \leftarrow \mathrm{y_c} + \Omega_{\mathrm{LENGTH}}[4]$;
20. **return with** $[\mathrm{x_{min}}, \mathrm{x_{max}}, \mathrm{y_{min}}, \mathrm{y_{max}}]$.

Algorithm 2 Corridor Generation

Function GenerateCorridors(χ_P)

1. $[\chi_{P_\mathrm{f}}, \chi_{P_\mathrm{r}}] \leftarrow$ GenerateReferenceTrajectory(χ_P);
2. $\Gamma \leftarrow \emptyset$;
3. **for** $k = 1 \ldots \mathrm{N_{FE}}$ **do**
4. $[\mathrm{xr}^k_{\mathrm{min}}, \mathrm{xr}^k_{\mathrm{max}}, \mathrm{yr}^k_{\mathrm{min}}, \mathrm{yr}^k_{\mathrm{max}}] \leftarrow$ GenerateLocalBox($\chi_{P_\mathrm{r}}[k].x, \chi_{P_\mathrm{r}}[k].\mathrm{y}$);
5. $\Gamma \leftarrow \Gamma \cup [\mathrm{xr}^k_{\mathrm{min}}, \mathrm{xr}^k_{\mathrm{max}}, \mathrm{yr}^k_{\mathrm{min}}, \mathrm{yr}^k_{\mathrm{max}}]$;
6. $[\mathrm{xf}^k_{\mathrm{min}}, \mathrm{xf}^k_{\mathrm{max}}, \mathrm{yf}^k_{\mathrm{min}}, \mathrm{yf}^k_{\mathrm{max}}] \leftarrow$ GenerateLocalBox($\chi_{P_\mathrm{f}}[k].x, \chi_{P_\mathrm{f}}[k].\mathrm{y}$);
7. $\Gamma \leftarrow \Gamma \cup [\mathrm{xf}^k_{\mathrm{min}}, \mathrm{xf}^k_{\mathrm{max}}, \mathrm{yf}^k_{\mathrm{min}}, \mathrm{yf}^k_{\mathrm{max}}]$;
8. **end for**
9. **return with** Γ.

D. Within-Corridor Constraint Formulation

With the two constructed corridors at hand, we formulate the within-corridor constraints, which are designated to replace the nominal collision-avoidance constraints (8) and (9).

The within-corridor constraints require that the points P_f and P_r should stay in the corresponding local boxes at specified time instances:

$$\begin{aligned} &P_\mathrm{f}(t) \text{ stays in} k\text{th local box for } P_\mathrm{f}, \quad \text{and} \\ &P_\mathrm{r}(t) \text{ stays in} k\text{th local box for } P_\mathrm{r}, \\ &t = T/\mathrm{N_{FE}} \cdot k, k = 1, \ldots, \mathrm{N_{FE}}. \end{aligned} \tag{14}$$

Since each local box is regularly placed, (14) is explicitly written as

$$x_\mathrm{f}(t) \in \left[\mathrm{xf}^k_{\mathrm{min}}, \mathrm{xf}^k_{\mathrm{max}}\right], \quad y_\mathrm{f}(t) \in \left[\mathrm{yf}^k_{\mathrm{min}}, \mathrm{yf}^k_{\mathrm{max}}\right],$$

$$x_{\mathrm{r}}(t) \in \left[\mathrm{xr}_{\min}^{k}, \mathrm{xr}_{\max}^{k}\right], \quad y_{\mathrm{r}}(t) \in \left[\mathrm{yr}_{\min}^{k}, \mathrm{yr}_{\max}^{k}\right], \quad t = T/\mathrm{N_{FE}} \cdot k, \quad k = 1, \ldots, \mathrm{N_{FE}}. \tag{15}$$

The within-corridor constraints (15) are box constraints w.r.t. $x_{\mathrm{f}}(t)$, $y_{\mathrm{f}}(t)$, $x_{\mathrm{r}}(t)$, and $y_{\mathrm{r}}(t)$. Since box constraints are the simplest type of linear constraints, replacing (8) and (9) with (12) and (15) makes the entire OCP tractably scaled and less complex. Now the modified OCP is presented as

$$\begin{aligned} &\min_{\mathbf{z}^*(t),\mathbf{u}(t),T} (10), \\ &\quad \text{s.t. Kinematic constraints (2), (3) and (12);} \\ &\qquad \text{Boundary conditions (4) and (5);} \\ &\qquad \text{Within-corridor constraints (15).} \end{aligned} \tag{16}$$

Herein, $\mathbf{z}^*$ is defined as the expanded state profile, i.e., $[\mathbf{z}, x_{\mathrm{f}}, y_{\mathrm{f}}, x_{\mathrm{r}}, y_{\mathrm{r}}]$; the geometric relation equation (12) is taken as part of the interior kinematic principle of the ego vehicle. In contrast with (11), the dimension of (16) is fully irrelevant to the complexity of the environment, which means that the number of constraints in (16) does not alter no matter how large $\mathrm{N_{OBS}}$ is. By solving (16) numerically, collocation points that represent a parking trajectory are derived.

IV. Lightweight Iterative Optimization Framework

A. Motivations

The preceding section has presented how to simplify the nominal OCP (11) into an easier version (16). However, (16) is subject to the following issues. First, the identified corridors inevitably leave some corner regions in the free space uncovered (see the blue regions in Fig. 6a). Second, the quality of the coarse trajectory (derived by FTHA introduced in Section III.B) may not be high, which influences the construction of the corridors. Due to the first issue, solving (16) renders a near-optimal rather than an optimal solution when the optimum falls into the uncovered regions. The second issue may make (16) fail to be solved if the constructed corridors do not cover any kinematically feasible trajectory. To summarize, directly solving (16) is not always reliable.

B. Iterative Optimization Framework

To address the issues pointed out in the preceding subsection, a natural idea is to adjust the corridors adaptively if they are found to be inappropriate. We establish an iterative framework, in which the corridors are built once per iteration and an intermediate OCP is formulated accordingly. The solution to the intermediate OCP in one iteration is used as the coarse trajectory χ_P for re-constructing the corridors in the next iteration. The entire iterative optimization process is initialized by the proposed FTHA algorithm, which guarantees to have an output within a limited time.

Intuitively, the iterative framework would work well to gradually find an optimal parking trajectory. However, one needs to ensure that each intermediate OCP is always solved with success, otherwise, an intermediate failure blocks any further evolution. To fix this underlying issue, we simplify (16) so that it never becomes infeasible. One may notice that the only nonlinear constraints in (16) include the kinematic equalities (2) and (12), as well as the end-point boundary condition (5b). If these nonlinear equalities are softened via external penalty functions and merged into the cost function J, (16) becomes a problem with purely box constraints. Eq. (2) is softened as

$$J_{\mathrm{EQ2}} = \int_{\tau=0}^{T} \| f\left(\mathbf{z}(\tau), \mathbf{u}(\tau)\right)\|^2 \cdot \mathrm{d}\tau. \tag{17}$$

Herein, $\| f\left(\mathbf{z}(\tau), \mathbf{u}(\tau)\right)\|^2$ measures the violation degree of $f\left(\mathbf{z}(\tau), \mathbf{u}(\tau)\right) = 0$ at $t = \tau$, which is integrated from 0 to T so as to form J_{EQ2}. Eq. (12), abstracted as $g\left(\mathbf{z}^*(t)\right) = 0$, is softened as

$$J_{\mathrm{EQ12}} = \int_{\tau=0}^{T} \left\| g\left(\mathbf{z}^*(\tau)\right)\right\|^2 \cdot \mathrm{d}\tau. \tag{18}$$

The end-point boundary condition (5b) is softened as

$$J_{\mathrm{EQ5B}} = \left(\sin\theta(T) - \sin(\theta_{\mathrm{F}})\right)^2 + \left(\cos\theta(T) - \cos(\theta_{\mathrm{F}})\right)^2. \tag{19}$$

The aforementioned three terms are summed up as $\psi_{\mathrm{infeasibility}}$:

$$\psi_{\mathrm{infeasibility}} = J_{\mathrm{EQ2}} + J_{\mathrm{EQ12}} + J_{\mathrm{EQ5B}}. \tag{20}$$

For a candidate solution, $\psi_{\mathrm{infeasibility}} = 0$ iff the constraints (2), (12), and (5b) are strictly satisfied. The compound cost function is presented as

$$J_{\mathrm{INT}} = J + \mathrm{w_{penalty}} \cdot \psi_{\mathrm{infeasibility}}, \tag{21}$$

wherein J is defined in (10), and $\mathrm{w_{penalty}} > 0$ should be set relatively large to ensure that the penalty for the violations of softened nonlinear constraints is dominant. But please note that setting $\mathrm{w_{penalty}}$ too high results in an ill-conditioned optimization problem.

The complete intermediate OCP is presented in the following form:

$$\begin{aligned} &\min_{\mathbf{z}^*(t),\mathbf{u}(t),T} (21), \\ &\quad \text{s.t. Kinematic constraints (3);} \\ &\qquad \text{Boundary conditions (4) and (5a);} \\ &\qquad \text{Within-corridor constraints(15).} \end{aligned} \tag{22}$$

Theoretically, (22) is never subject to the risk of being infeasible because every vector that lies between the box boundaries would be a feasible solution that satisfies all of the box constraints. As (22) does not contain complex constraints, its numerical solution process is extremely fast.

We use the following pseudo-codes to present our proposed autonomous parking trajectory planning method.

The inputs of ParkingTrajectoryPlanning() include *map* and *task*. *map* presents the environmental setup, i.e. the geometric size and location of each obstacle, as well as the scenario boundaries. *task* presents the parking scheme, i.e. the initial and terminal configurations of the ego vehicle. The output of the function is the optimized parking trajectory in the form of a sequence of timed configuration points $\chi_P = \left\{\sigma_0, \sigma_1, \ldots, \sigma_{N_{FE}}\right\}$. In line 1 of Alg. 3, GenerateCoarseTrajectoryViaFTHA() is used to generate a coarse trajectory. In line 2, FormInitialGuess() approximates the control profiles and the

Algorithm 3 Autonomous Parking Trajectory Planning

Function ParkingTrajectoryPlanning($map, task$)

1. $\chi_P \leftarrow$ GenerateCoarseTrajectoryViaFTHA($map, task$);
2. $sol \leftarrow$ FormInitialGuess(χ_P);
3. $iter \leftarrow 0$, $\psi_{\text{infeasibility}} \leftarrow +\infty$;
4. $dilated\_map \leftarrow$ DilateMap(map);
5. **while** $\psi_{\text{infeasibility}} \geq \varepsilon_{\text{tol}}$ **do**
6. $iter \leftarrow iter + 1$;
7. **if** $iter >$ $\text{iter}_{\max}$ **then**
8. **return** with failure;
9. **end if**
10. $\Gamma =$ GenerateCorridors(χ_P);
11. $OCP_{\text{INT}} \leftarrow$ GenerateOCP($task, \Gamma$);
12. $sol \leftarrow$ SolveOCP(OCP_{INT}, sol);
13. $\psi_{\text{infeasibility}} \leftarrow$ MeasureInfeasibility(sol);
14. $\chi_P \leftarrow$ ExtractTrajectory(sol);
15. **end while**
16. **return with** χ_P.

rest state profiles based on χ_P to form an initial guess. In line 4, the function DilateMap() is deployed to inflate the original obstacles so that a dilated map is built for corridor construction. In line 5, $\varepsilon_{\text{tol}} \to 0^+$ is a user-specified parameter denoting the convergence acceptance threshold, i.e., the softened nonlinear constraints are regarded as satisfied if $\psi_{\text{infeasibility}} < \varepsilon_{\text{tol}}$. The parameter $\text{iter}_{\max}$ in line 7 denotes the maximum allowable iteration number, and exceeding this number renders a solution failure. GenerateOCP() formulates an intermediate OCP in the form of (22). SolveOCP(OCP_{INT}, sol) stands for solving OCP_{INT} numerically. Concretely, it is about converting OCP_{INT} into an NLP problem and solving it via an NLP solver, which is warm-started by sol. With a solution vector sol at hand, we calculate its infeasibility degree $\psi_{\text{infeasibility}}$ via MeasureInfeasibility() according to (20). ExtractTrajectory() is a simple function to extract the trajectory part from the entire solution vector.

Remark 1: The final output of Alg. 3 is feasible w.r.t. the nominal planning scheme (11).

Recall that the output of Alg. 3, denoted as χ_P, is derived by solving an intermediate OCP in the form of (22). Thus χ_P satisfies all of the constraints in (22), including the boundary conditions (4) and (5a), and kinematics-related box constraints (3).

According to the criterion to exit the while loop, the finally derived χ_P satisfies that $\psi_{\text{infeasibility}} < \varepsilon_{\text{tol}}$. According to the semi-positive definitions in (17)–(20), $\psi_{\text{infeasibility}} < \varepsilon_{\text{tol}}$ indicates that

$$J_{\text{EQ2}} < \varepsilon_{\text{tol}}, \quad J_{\text{EQ12}} < \varepsilon_{\text{tol}}, \quad J_{\text{EQ5B}} < \varepsilon_{\text{tol}}. \tag{23}$$

When $\varepsilon_{\text{tol}} \to 0^+$, J_{EQ2}, J_{EQ12}, and J_{EQ5B} approaches 0^+ as well, which means the constraints (2), (12), and (5b) are satisfied.

Besides that, the usage of two disks to cover the rectangular vehicle body, the establishment of two tunnels, and the satisfaction of the within-tunnel constraints (15) ensure that χ_P satisfies the nominal collision-avoidance constraints (8) and (9). Since χ_P does not violate any of the constraints in the nominal OCP (11), Alg. 3 always provides a feasible solution to it.

Remark 2: The final output of Alg. 3 is near-optimal w.r.t. the nominal planning scheme (11) if $\text{w}_{\text{penalty}}$ is sufficiently large.

If we set $\text{w}_{\text{penalty}} \gg \varepsilon_{\text{tol}}^{-1}$, then $\text{w}_{\text{penalty}} \cdot \psi_{\text{infeasibility}} \to 0^+$ w.r.t. the final output χ_P. In that case, the intermediate cost function $J_{\text{INT}} \to J$, which means the cost function of the last intermediate OCP is consistent with the nominal OCP (11).

Let us denote the feasible region of an OCP as Γ_{ID}, where *ID* denotes the equation index. Γ_{22} is a subset of Γ_{11} because the establishment of the within-corridor constraints defiantly indicates that the vehicle body is presented by two dicks rather than a rectangle. The usage of two discs wastes free spaces inevitably. Since Γ_{22} is slightly smaller than Γ_{11}, χ_P is not an optimal solution if the optimum lies in $\Gamma_{22} \cap^{\neg} \Gamma_{11}$.

TABLE I

PARAMETRIC SETTINGS REGARDING MODEL AND APPROACH

Parameter	Description	Setting
L_{F}	Front hang length of vehicle	0.96 m
L_{W}	Wheelbase of vehicle	2.80 m
L_{R}	Rear hang length of vehicle	0.929 m
L_{B}	Width of vehicle	1.942 m
$\text{a}_{\max}$	Upper bound of $\lvert a(t)\rvert$	0.4 m/s^2
$\text{v}_{\max}$	Upper bound of $\lvert v(t)\rvert$	2.5 m/s
$\Phi_{\max}$	Upper bound of $\lvert \phi(t)\rvert$	0.7 rad
$\Omega_{\max}$	Upper bound of $\lvert \omega(t)\rvert$	0.5 rad/s
$\text{w}, \text{w}_{\text{penalty}}$	Weights in cost functions (10) and (21)	10^{-2}, 10^{9}
N_{FE}	Number of sampled waypoints	50
Δs , L_{limit}	Unit / Maximum step length in Alg. 3	0.1 m, 10 m
ε_{tol}	Convergence tolerance in Alg. 3	10^{-6}
$\text{iter}_{\max}$	Maximum iteration number to terminate the iterations in Alg. 3	10

V. EXPERIMENTAL SETUPS, RESULTS, AND DISCUSSIONS

A. Simulation Setup

Simulations were conducted in C++ and executed on an i9-9900 CPU with 32 GB RAM that runs at 3.10 × 2 GHz. Regarding the function SolveOCP() defined in Section IV.B, we use the first-order explicit Runge-Kutta method to form the NLP problem and use the primal-dual interior-point solver IPOPT [37] with the linear solver MA97 [38] called through the AMPL interface [39]. Basic parametric settings are listed in Table I.

A set containing 115 simulation cases is built to evaluate the performance of the proposed parking trajectory planner. Manually selected from 6,000 randomly generated cases, the 115 cases are classified as maneuvering, detouring, and narrow passage traverse. Each case contains 5 polygonal obstacles; the vertex number of each obstacle randomly ranges from 4 to 7; the location of each vertex obeys the uniform distribution within a 40m × 40m workspace; each polygonal obstacle's area ranges randomly from 5m^2 to 50m^2; the initial

TABLE II
DEFINITIONS OF COMPARABLE TRAJECTORY PLANNERS

Algorithm ID	Description
Naïve NLP	Solve the nominal OCP (11) numerically [18], where the initial guess is provided by FTHA.
STC	Numerically solve the OCP (16) only once [30], where the initial guess is provided by FTHA.
TEB	Timed elastic bound method proposed in [25], where the initial guess is provided by FTHA (the maximum iteration number in the outside loop is set to 500).
TDR-OBCA	A temporal and dual warm starts with reformulated optimization-based collision avoidance method proposed in [40]

TABLE III
COMPARATIVE SIMULATION RESULTS

Alg. ID	Success rate	Average CPU time / s	99% percentile CPU time / s
This work	100.00%	0.909	3.138
Naïve NLP	91.30%	3.255	9.193
STC	96.52%	0.763	1.206
TEB	65.22%	0.278	2.896
TDR-OBCA	100.00%	2.048	4.014

and terminal locations of the parking vehicle are uniformly distributed in the workspace; the initial and goal orientations are uniformly distributed in $[-2\pi, 2\pi]$; and the other boundary constraints are set to

$$\begin{bmatrix} v(0) \\ a(0) \\ \phi(0) \\ \omega(0) \end{bmatrix} = \begin{bmatrix} v(T) \\ a(T) \\ \phi(T) \\ \omega(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

B. Simulation Results and Discussions

To evaluate the performance of the proposed trajectory planner, comparable algorithms are defined in Table II. The performances of Alg. 3 and its competitors on the 115 simulation cases are summarized in Tables III.

Directly solving the nominal OCP (11) is time-consuming. Given that the nominal collision-avoidance constraints (8) and (9) are highly non-convex and almost non-differentiable, the Naïve NLP solution process cannot easily get correct gradient information when the initial guess is not close to being optimal. As the result, 10 of the gross 115 cases fail to be solved by the Naïve NLP method.

The STC method is featured by solving the OCP (16) once. Since the collision-avoidance constraints in the OCP (16) are linear, the numerical solution process is faster. However, as we mentioned in Section IV.A, the STC method relies on the initial guess. In each of the 4 failed cases, the corridor construction operation leaves out essential free spaces to form a feasible solution.

Regarding the TEB method, the nominal OCP is softened as an unconstrained optimization problem and then solved quickly via the g^2o solver. TEB is featured by running fast, but the finally derived solution may violate the kinematic constraints or the collision-avoidance constraints, which is regarded as a solution failure.

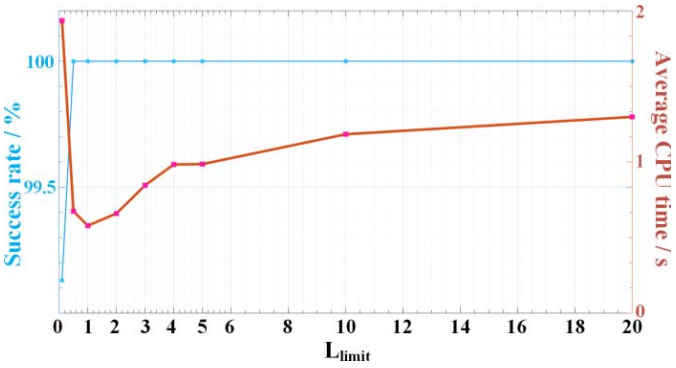


Fig. 7. On the performance of Alg. 3 under different settings of L_{limit}.

The TDR-OBCA method is a variant of the conventional OBCA method [20]. In contrast with the OBCA method, the TDR-OBCA method refines the coarse trajectory before using it as the initial guess. Also, it reformulates the constraints/cost function to enhance the planning robustness and efficiency. As seen in Table III, all of the 115 cases are solved with success using the TDR-OBCA method, but the CPU runtime is longer because that method, like the basic OBCA method, does not consider the intractability issue of the collision-avoidance constraints.

Our proposed planning method scales tractably w.r.t the environmental obstacles, and is insensitive to the quality of the initial guess at the very beginning. It builds a lightweight iterative framework to recover from kinematical infeasibility. We name the framework as lightweight because each intermediate OCP only contains a tractable number of box constraints. These are the reasons that our proposal outperforms its competitors listed in Table II.

In our proposed Alg. 3, L_{limit} is a critical parameter that decides the maximum length/width of each local box. By setting L_{limit} to different values, the changes in the success rate and the average CPU time are depicted in Fig. 7. From that figure, one may conclude that setting L_{limit} smaller makes the corridor construction faster but leads to more iterations; setting L_{limit} larger consumes more time in constructing the corridors within one iteration, but fewer iterations are needed before the trajectory planning process is completed. The success rate is not affected unless L_{limit} is set overly small ($L_{limit} = 0.1$). This is a merit of our proposed iterative framework. As a comparison, in using the STC method to solve the 115 cases with $L_{limit} = 0.1$, we find that the success rate drastically reduces to 6.09%.

A video containing the simulation results is available at bilibili.com/video/BV1n7411q7iv/. A typical simulation result is depicted in Fig. 8.

C. Real-World Experimental Setups and Results

Besides simulations, we have also carried out real-world experiments based on a small car-like robot in a 1.8m $\times$1.2 m indoor scenario.

As depicted in Fig. 9, the vehicle localization is done visually. Concretely, the location and orientation of the car-like robot are identified through tracking the AprilTag placed on the top of the robot via a bird-eye-view camera. AprilTags are

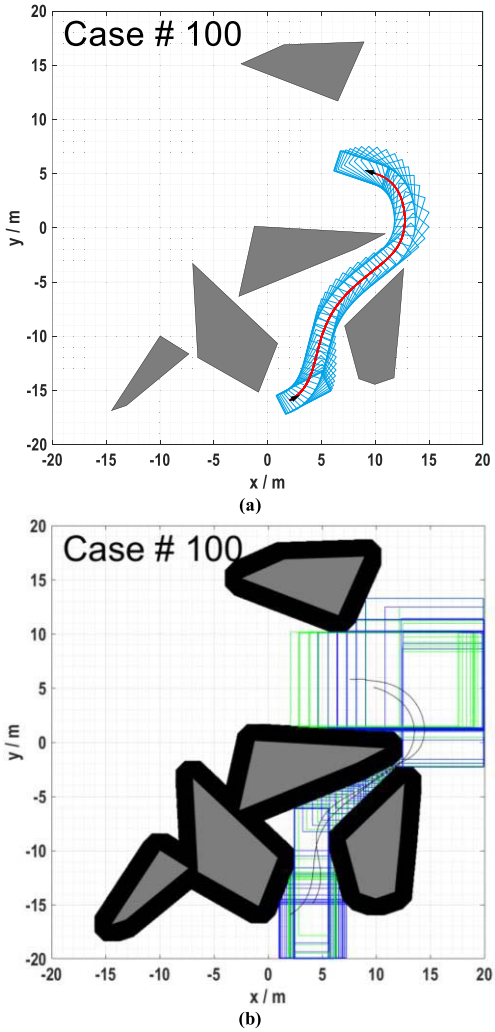


Fig. 8. Typical simulation results: (a) footprints of planned parking trajectory; (b) constructed corridors consisting of two series of local boxes.

Fig. 9. Experimental platform based on an ackermann-steering car-like robot.

also used to "percept" the location and orientation of each registered obstacle. The perception and localization information is sent to a desktop PC, in which our proposed autonomous parking trajectory planning method would be implemented once to generate an open-loop trajectory before the robot begins to move. In using the proposed planner, we add a buffer of 0.01m to the radius of each covering disc such that collisions caused by small tracking errors would be

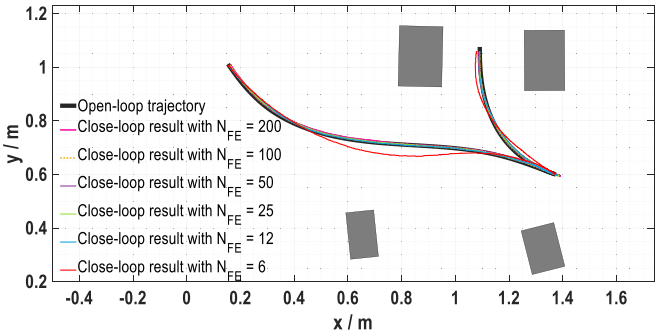


Fig. 10. Closed-loop parking trajectory tracking performances under various discretization accuracy conditions.

TABLE IV
UPDATED PARAMETRIC SETTINGS FOR EXPERIMENTS

Parameter	Setting	Parameter	Setting
L_F	0.034 m	L_W	0.143 m
L_R	0.034 m	L_B	0.191 m
a_{max}	0.05 m/s^2	v_{max}	0.2 m/s
Φ_{max}	0.3491 rad	Ω_{max}	0.1 rad/s
Δs	0.005 m	L_{limit}	2.0 m

avoided [41]. Parametric settings different from Table I are listed in Table IV. The derived open-loop trajectory is sent to the robot platform via ZigBee communication. Thereafter, the open-loop trajectory is used for closed-loop control. Particularly in the closed-loop trajectory control module, we use a PID controller for longitudinal tracking and use a pure pursuit controller for lateral tracking. We set the execution frequency of the control module to 10.0 Hz.

Fig. 10 shows the closed-loop tracking performances under various conditions in a typical vertical parking scenario. We generate an extremely precise open-loop trajectory under $N_{FE} = 200$ (N_{FE}, as listed in Table I, decides the discretization accuracy when converting an OCP to an NLP problem). Thereafter, the open-loop trajectory is down-sampled by setting N_{FE} to smaller values and then sent to the low-level controller for closed-loop tracking. The results show that the numerical discretization accuracy is not a dominant factor that influences the whole parking performance. Also, Fig. 10 shows that our planner can generate easy-to-track trajectories.

VI. CONCLUSION

This paper has proposed an autonomous parking trajectory planning method in a generic unstructured environment. To get precise and optimal solutions, we build an OCP to describe the concerned scheme. To solve it efficiently, we convert the nominal collision-avoidance constraints to within-corridor conditions via STC. The performance of the adopted STC method depends on the quality of the initial guess, thus we 1) propose the FTHA method to enhance the time efficiency of the coarse trajectory generation process, and 2) build an iterative framework to gradually recover from a poor initial guess.

Generally speaking, the optimization-based planners run more slowly than the sampling-and-search-based planners,

thus how to enhance the runtime efficiency is a primary concern. This work has confirmed that simply softening the hard constraints via external penalty functions does not guarantee solution feasibility, although it makes the optimization process faster. Linearizing the original collision-avoidance constraints without making their scale tractable, as the OCBA-related methods do, renders a low calculation speed. Relying highly on the initial guess, as the STC method does, is not a promising approach especially in dealing with intricate scenarios with narrow passages. By contrast, our proposed method balances the solution accuracy, optimality, solution speed, and robustness (weak reliance on the initial guess). All of the difficulties in the planning scheme can be dispersed into the intermediate problems defined in our iterative framework. We expect that the kinematics infeasibility and the solution optimality can evolve in cooperation during the iteration. In this sense, there is no need to make the intermediate procedures too strict, thus we make each intermediate problem lightweight, i.e., a simple optimization problem with fixed-scale box constraints. Comparative simulation results have shown that our proposal outperforms the other few optimization-based strategies w.r.t the CPU runtime and success rate. The proposed iterative framework can be extended to handle generic high-dimensional state transfer problems with nonlinear system dynamics and large-scale non-convex exterior constraints.

It deserves to point out that our proposed planner cannot alter the homotopy class after a coarse trajectory is identified at the very beginning via FTHA. If the coarse trajectory is homotopically misleading, then our proposed method is not able to find a different homotopy class. As our future work, fault-recovery strategies should be designed in the outer loop of our iterative framework so that imperfect homotopy classes can be altered, which makes the entire optimization-based planner insensitive to the initial guess.

ACKNOWLEDGMENT

Bai Li would like to thank Prof. Li Li, Dr. Weitian Sheng, Dr. Oskar Ljungqvist, Prof. Changliu Liu, Muyuan Niu, Peng Song, Yu Zheng, and Shiliang Jin for their support during this study.

REFERENCES

[1] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? An overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3135–3151, Aug. 2020.

[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. Autom. Lett.*, vol. 6, no. 3, pp. 4536–4543, Jul. 2021.

[3] L. Palmieri, L. Bruns, M. Meurer, and K. O. Arras, "Dispertio: Optimal sampling for safe deterministic motion planning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 362–368, Apr. 2020.

[4] 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.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[6] 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.

[7] C. Jang, C. Kim, S. Lee, S. Kim, S. Lee, and M. Sunwoo, "Re-plannable automated parking system with a standalone around view monitor for narrow parking lots," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 2, pp. 777–790, Feb. 2020.

[8] 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.

[9] 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.

[10] M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2010, pp. 3071–3077.

[11] 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.*, May 2011, pp. 5622–5627.

[12] 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. (ICRA)*, May 2016, pp. 2775–2781.

[13] 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.

[14] 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 (ICM)*, Mar. 2021, pp. 1–6.

[15] 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, Sep. 2015.

[16] P. Zips, M. Bock, and A. Kugi, "A fast motion planning algorithm for car parking based on static optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 2392–2397.

[17] K. Kondak and G. Hommel, "Computation of time-optimal movements for autonomous parking of non-holonomic mobile platforms," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 3, May 2001, pp. 2698–2703.

[18] 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, Sep. 2015.

[19] S. Shi, Y. Xiong, J. Chen, and C. Xiong, "A bilevel optimal motion planning (BOMP) model with application to autonomous parking," *Int. J. Intell. Robot. Appl.*, vol. 3, no. 4, pp. 370–382, Dec. 2019.

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

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

[22] B. Li, Y. Zhang, and Z. Shao, "Spatio-temporal decomposition: A knowledge-based initialization strategy for parallel parking motion optimization," *Knowl.-Based Syst.*, vol. 107, pp. 179–196, Sep. 2016.

[23] C. Sun, Q. Li, and L. Li, "A gridmap-path reshaping algorithm for path planning," *IEEE Access*, vol. 7, pp. 183150–183161, 2019.

[24] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, and Y. Xia, "Two-stage trajectory optimization for autonomous ground vehicles parking maneuver," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 3899–3909, Jul. 2019.

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

[26] M. Vitus, V. Pradeep, G. Hoffmann, S. Waslander, and C. Tomlin, "Tunnel-MILP: Path planning with sequential convex polytopes," in *Proc. AIAA Guid., Navigat. Control Conf. Exhib.*, Aug. 2008, p. 7132.

[27] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. 54th IEEE Conf. Decis. Control (CDC)*, Dec. 2015, pp. 835–842.

[28] 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. (ICRA)*, May 2015, pp. 662–669.

[29] C. Liu and M. Tomizuka, "Real time trajectory optimization for nonlinear robotic systems: Relaxation and convexification," *Syst. Control Lett.*, vol. 108, pp. 56–63, Oct. 2017.

[30] 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. Eur. Control Conf. (ECC)*, May 2020, pp. 1993–1998.

[31] T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl, "An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2020, pp. 3574–3580.
[32] 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. Vehicles*, vol. 5, no. 3, pp. 414–425, Sep. 2020.
[33] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "An admissible heuristic to improve convergence in kinodynamic planners using motion primitives," *IEEE Control Syst. Lett.*, vol. 4, no. 1, pp. 175–180, Jan. 2020.
[34] S. Liu *et al.*, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Feb. 2017.
[35] J. Park and H. J. Kim, "Online trajectory planning for multiple quadrotors in dynamic environments using relative safe flight corridor," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 659–666, Apr. 2021.
[36] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2010, pp. 518–522.
[37] 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, May 2006.
[38] HSL. *A Collection of Fortran Codes for Large Scale Scientific Computation*. [Online]. Available: http://www.hsl.rl.ac.uk/
[39] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*. San Francisco, CA, USA: Scientific Press, 2003.
[40] R. He *et al.*, "TDR-OBCA: A reliable planner for autonomous driving in free-space environment," 2020, *arXiv:2009.11345*. [Online]. Available: http://arxiv.org/abs/2009.11345
[41] B. Li, Y. Ouyang, Y. Zhang, T. Acarman, Q. Kong, and Z. Shao, "Optimal cooperative maneuver planning for multiple nonholonomic robots in a tiny environment via adaptive-scaling constrained optimization," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1511–1518, Apr. 2021.
[42] C. Rösmann, A. Makarow, and T. Bertram, "Online motion planning based on nonlinear model predictive control with non-Euclidean rotation groups," 2020, *arXiv:2006.03534*. [Online]. Available: https://arxiv.org/abs/2006.03534

Bai Li (Member, IEEE) received the B.S. degree from the School of Advanced Engineering, Beihang University, China, in 2013, and the Ph.D. degree from the College of Control Science and Engineering, Zhejiang University, China, in 2018. From November 2016 to June 2017, he visited the Department of Civil and Environmental Engineering, University of Michigan, Ann Arbor, MI, USA, as a Joint Training Ph.D. Student. Prior to teaching at Hunan University, China, he worked with the JDX Research and Development Center of Automated Driving, JD Inc., China, as an Algorithm Engineer, from 2018 to 2020. He is currently an Associate Professor with the College of Mechanical and Vehicle Engineering, Hunan University. He has been the first author of nearly 60 journals/conference papers and two books in the community of robotics. His research interest includes motion planning of automated vehicles. He was a recipient of the International Federation of Automatic Control (IFAC) 2014–2016 Best Journal Paper Prize.

Tankut Acarman (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2002. He is currently a Professor and the Head of the Department of Computer Engineering, Galatasaray University, Istanbul, Turkey. He is a coauthor of the book entitled *Autonomous Ground Vehicles*. His research interests include aspects of intelligent vehicle technologies, driver assistance systems, and performance evaluation of inter-vehicle communication.

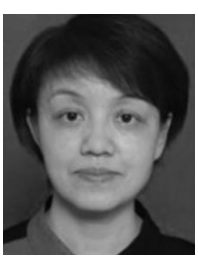
Youmin Zhang (Senior Member, IEEE) is currently a Professor with the Department of Mechanical, Industrial and Aerospace Engineering and the Concordia Institute of Aerospace Design and Innovation, Concordia University, Montreal, QC, Canada. He has authored eight books, over 550 journals and conference papers, and book chapters. His main research interests include guidance, navigation and control, fault detection and diagnosis, and fault-tolerant control of unmanned systems.

Yakun Ouyang (Student Member, IEEE) received the B.S. degree from the School of Information Engineering, Nanchang University, Nanchang, China, in 2020. He is currently pursuing the master's degree with the College of Mechanical and Vehicle Engineering, Hunan University, China. His research interests include decision making, and trajectory planning, control, and software engineering of an autonomous vehicle systems. He was a first-prize recipient of the 2019 National University Students Intelligent Car Race.

Cagdas Yaman received the B.S. and M.S. degrees in computer engineering from Galatasaray University, Turkey, in 2011 and 2016, respectively, where he is currently pursuing the Ph.D. degree. His research interests include vehicular communication and autonomous vehicle-related technologies.

Qi Kong (Member, IEEE) received the B.S. and M.S. degrees in computer science from Shanghai Jiaotong University, China. In 2018, he joined as the Chief Scientist of autonomous driving to lead the Autonomous Driving Department, JD.com, both in China and America. Prior to JD.com, he was a Senior Principal Architect at Baidu USA and one of the Founding Member of Baidu Apollo Project. He has presided over several industrial projects in the past ten years. His research interests include autonomous driving, big data, and machine learning.

Xiang Zhong received the B.S. degree from the Department of Electrical Engineering, Hunan University, in 1993, and the M.S. degree from the School of Software, Hunan University, in 2014. He is currently an Assistant Professor with the College of Mechanical and Vehicle Engineering, Hunan University, the President of Hunan Intelligent Transportation Industry Association, and the Deputy Director of Hunan Provincial Key Laboratory of Big Data Research and Applications. His current research interests include intelligent transportation, traffic big data analysis, and vehicle road collaboration.



Xiaoyan Peng received the B.S. and M.S. degrees in mechanical engineering, and the Ph.D. degree in automatic control from Hunan University, Changsha, China, in 1986, 1989, and 2013, respectively. She is currently a Professor with the College of Mechanical and Vehicle Engineering, Hunan University. Her research interests include control of mechatronic systems and safety analysis of autonomous vehicles.