Fast Optimization-Based Trajectory Planning with Cumulative Key Constraints for Automated Parking in Unstructured Environments

Zijun Guo, Yuanxin Wang, Huilong Yu, *Member, IEEE*, and Junqiang Xi, *Member, IEEE*

***Abstract*—The optimization problem for trajectory planning becomes intractable as the dimensions of collision avoidance constraints increase. Existing methods either avoid unstructured environments or use simplified constraints that sacrifice a portion of the solution space. To tackle the intractability while preserving the feasible region, we introduce trajectory planning with cumulative key constraints (TPCKC), with which we won first prize in the trajectory planning competition of automated parking (TPCAP). In the proposed method, only the violated vertex-to-polytope constraints are treated as key constraints and added to a collision avoidance constraint set. Iteratively, an optimization problem with the constraint set is solved, and its solution is checked for new collisions. The cumulation of constraints ends when the solution, restricted by key constraints only, is collision-free. The proposed method is compared with three optimization-based representatives on the TPCAP benchmarks. Practical real-time performance in all tested cases, together with the highest success rate and trajectory quality, is achieved with the proposed method. Besides simulation, TPCKC is also validated in a real-world experiment on an electric chassis platform under environmental changes.**

***Index Terms*—Automated parking, motion planning, trajectory planning, obstacle avoidance, nonlinear programming.**

I. Introduction

A. Background

MOTION planning in autonomous driving serves as a bridge between behavioral decision making and vehicle control [1], [2]. A typical application of motion planning is automated parking, the goal of which is to navigate the vehicle from its start pose to an end pose amid obstacles [3]. The parking environments are perceived as unstructured or poorly structured when the obstacles are cluttered [4] or irregularly placed [5], as illustrated in Fig. 1. These obstacles weaken the road structures and narrow the drivable areas, which makes planning more challenging [6]. Confronted with unstructured environments, search-based [7] and sampling-based [8] methods are proposed to compute collision-free paths. However, these path planning techniques focus on feasibility instead of optimality [9], resulting in coarse paths difficult for low-level controllers to follow. Aided by optimization, trajectory planning techniques that generate not only paths but also corresponding vehicle states improve the drivability of the planning results [10], [11]. The downside to optimization is higher computational time, mainly due to the nonlinear constraints for boundary conditions, vehicle kinematics, and collision avoidance [12], [13].

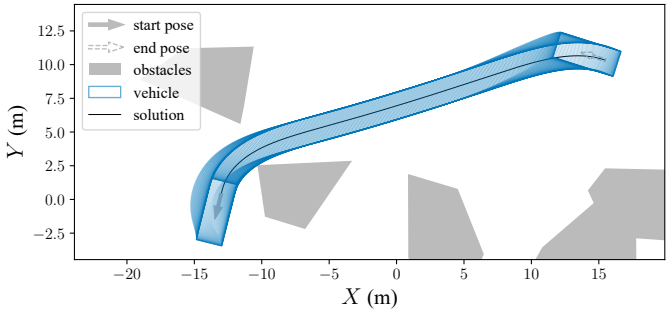

Fig. 1. A simple example of unstructured parking environments (Case 11 in [14]). The trajectory is planned with the method proposed in this work. Note that the vehicle and obstacles in the visualization may overlap slightly due to the existence of line width.

B. Related Works

The methods for optimization-based trajectory planning in unstructured environments can be roughly divided into two categories: methods that preserve the feasible region and methods that sacrifice a portion of the feasible region.

In the first category, mixed integer problems or nonlinear constraints are formulated to describe collision avoidance exactly. In [15], the feasible region is divided into a sequence of convex polytopes, and a mixed integer linear programming problem is formulated. One and a half minutes is required to solve in a 20-obstacle environment. With no integer variables, a unified optimization method is proposed in [16] using triangle area criterion. However, the calculation of area involves taking absolute values, which is non-differentiable and hard to solve. To address this issue, a group of methods called Optimization-Based Collision Avoidance (OBCA) [11], [17], [18] provides smooth formulations of distance and signed distance between two convex sets aided by auxiliary dual variables. Problems of reverse and parallel parking with less than 5 regularized obstacles can be solved within seconds, which is perceived as real-time performance in practice. However, fast solving in unstructured environments still cannot be guaranteed, which owes to the intractability caused by the

This work was supported by the National Natural Science Foundation of China under Grant 52472444. *(Corresponding author: Huilong Yu.)*

The authors are with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China (e-mail: zijun.guo@bit.edu.cn; yuanxin.wang@bit.edu.cn; huilong.yu@bit.edu.cn; xijunqiang@bit.edu.cn).

increased number of nonlinear constraints. Although different initialization strategies [19], [20] are proposed to alleviate the intractability, the planning time is still relatively high.

In the second category, the nonlinearity in collision avoidance constraints is reduced or transformed to convexity at the cost of a portion of the feasible region. To compensate for the cost, methods in this category are usually iterative. The signed distance function is linearized, and a sequential convex optimization approach is adopted in [21] for the motion planning of a wide range of robots. Later, an iterative algorithm using the convex feasible set, a convex corridor from the start pose to the end pose, is proved to guarantee local minima and used on mobile robots [22]. Iteratively, a convex feasible set is found, and a convex optimization problem is solved until convergence. The aforementioned corridor method is applied to automated parking in unstructured environments in [5], where the rectangular vehicle is approximated by two disks and two box-like corridors are constructed at every iteration to restrict the disk centers from collision with dilated obstacles. With the feasible region truncated, the trajectory quality of the above methods can hardly surpass that of methods in the first category. In some tight cases, the above methods may fail due to the geometric approximation of the vehicle shape. While these drawbacks are attributed to the corridor methods, the knowledge-based simplification of collision avoidance constraints seems promising. In [23], the vehicle is restricted to a convex collision-free region containing the end pose in the last period of the parking maneuver, resulting in an easy-to-solve problem. Many extremely tight problems considered infeasible by methods in the first category can be solved successfully. However, this method is not general and relies on prior knowledge.

The proposed method in this work, although iterative, belongs to the first category where the feasible region is preserved. Compared with the previous ones in this category, our method solves significantly faster by enforcing fewer and simpler constraints and generates trajectories of higher quality due to reduced nonlinearity. Like the others in the first category, our method outperforms the general methods in the second category in trajectory quality. The reasons have been discussed above.

C. Motivations

It is redundant to enforce collision avoidance constraints with every obstacle in the environment at every discrete time, as faraway obstacles have no chance of collision. We take steps further by arguing that 1) even in close vicinity, obstacles on one side may be omitted when the vehicle tends to knock into those on the other side; 2) some vertex-to-polytope constraints may be omitted when the chances of collision are slim. The above arguments suggest that we impose only key constraints. But how to identify them?

In this work, we treat only violated constraints as key constraints. Suppose we solve for a time-optimal trajectory around a search-based initial guess without considering obstacles, chances are that the solution is collision-free. If not, the collision information should be collected, and the corresponding constraints, i.e. key constraints, should be enforced. As new collisions may arise after the next problem-solving, the process should iterate, and the key constraints should cumulate until the final solution becomes collision-free. A video for algorithm visualization is available in the supplementary material.

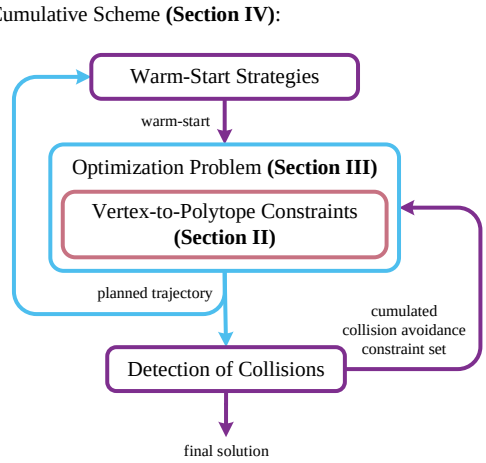

Fig. 2. Illustration for the relationship of Section II, III, and IV.

D. Contributions

The contributions of this work are summarized as follows.

1) We formulate two kinds of vertex-to-polytope constraints for collision avoidance to substitute the complete constraints between two convex sets. The formulation is smooth, downscaled, and targeted at vertex intrusion.
2) Instead of imposing all collision avoidance constraints, we only enforce key constraints in the optimization problem. We show that the reduction of constraint dimensions improves both the computational speed and trajectory quality.
3) We propose a cumulative scheme, which is able to capture key collision avoidance constraints in few iterations. The scheme is compared on 18 cases selected from the TPCAP benchmarks [14] and validated in a real-world experiment.

E. Organization

The remainder of this work is organized as follows. Section II formulates the vertex-to-polytope collision avoidance constraints. Section III describes the optimization problem compatible with the proposed constraints and scheme in Section II and IV. Section IV elaborates on the cumulative scheme to capture key constraints. The relationship of Section II, III, and IV are depicted in Fig. 2. Section V performs experiments and analyzes results. Section VI concludes this work.

II. Vertex-to-Polytope Constraints

In this section, two kinds of vertex-to-polytope (V2P) constraints for collision avoidance are formulated, replacing the complete constraints between two convex sets implemented in OBCA methods [11], [17], [18]. The change from OBCA

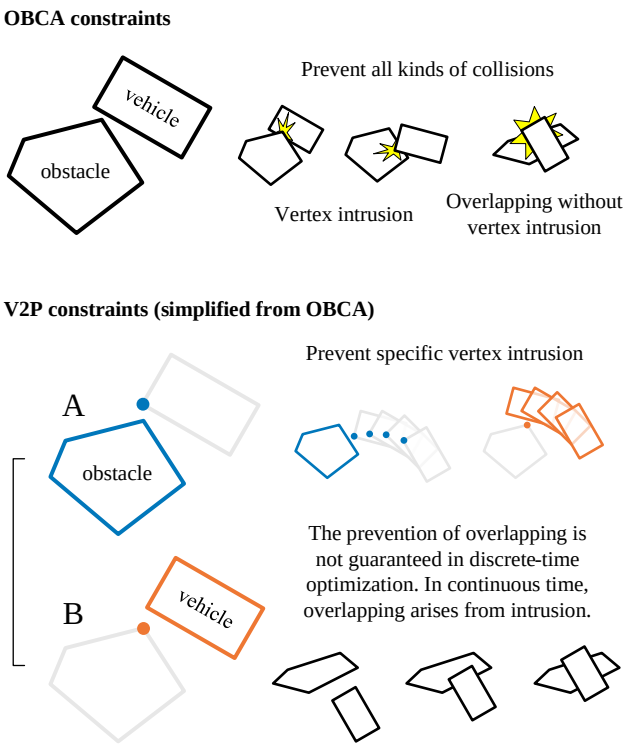

Fig. 3. Illustration for the change of collision avoidance constraints (top: OBCA, bottom: V2P). A and B correspond to the formulations in Section II-A and Section II-B, respectively. The V2P constraints can be applied to any vertex. Vertices in the figure are chosen for illustrative purposes.

to V2P, illustrated in Fig. 3, is a simplification that enlarges the feasible region. The reasons are twofold: 1) only vertex intrusion is considered, neglecting the overlapping of convex polytopes without vertex intrusion; 2) vertex intrusion can be tackled directly without redundancy, which makes it possible to leave out unnecessary constraints.

The V2P formulation is applicable if the vehicle and obstacle shapes are known and can be approximated by polytopes. In practice, it can be achieved with sophisticated perception techniques, e.g. point cloud segmentation and convex hull computation [24]. The polytopes, if non-convex, should be decomposed into convex ones, e.g. by using [25]. In this work, it is assumed that all the obstacles are static.

A. Constraints Between a Vehicle Vertex and an Obstacle

Convex polytopes can be properly expressed using the half-space representation. A 2D convex-polytope obstacle with nonempty relative interior is given by

$$\mathbb{O} = \left\{ \boldsymbol{q} \in \mathbb{R}^2 : \boldsymbol{A}\boldsymbol{q} \preceq \boldsymbol{b} \right\}, \tag{1}$$

where $\boldsymbol{q}$ is a point in the convex polytope, $\boldsymbol{A} \in \mathbb{R}^{l\times 2}$ and $\boldsymbol{b} \in \mathbb{R}^l$ are parameterized by the obstacle edges, and l corresponds to the number of edges. In this work, $\preceq$ and $\succeq$ are used to express element-wise inequalities $\leq$ and $\geq$, respectively, and $\mathbf{0}$ represents a vector of zeros.

For a given vehicle vertex $\boldsymbol{p}(\boldsymbol{x}) \in \mathbb{R}^2$, where $\boldsymbol{x}$ represents the vehicle states, the distance between the vertex and the obstacle $\text{dist}(\boldsymbol{p}(\boldsymbol{x}), \mathbb{O})$ is obtained by first finding a point $\boldsymbol{q}$ in the obstacle $\mathbb{O}$ that is closest to $\boldsymbol{p}(\boldsymbol{x})$, then calculating the distance between $\boldsymbol{p}(\boldsymbol{x})$ and $\boldsymbol{q}$. Therefore, the distance is the optimal value of the following optimization problem [26, Section 8.1]:

$$\begin{aligned} \min_{\boldsymbol{q}} \quad & \|\boldsymbol{s}\| \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{q} \preceq \boldsymbol{b} \\ & \boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{q} = \boldsymbol{s}, \end{aligned} \tag{2}$$

where $\|\cdot\| = \|\cdot\|_2$ is the Euclidean norm, $\boldsymbol{q}$ is a point in the obstacle $\mathbb{O}$, $\boldsymbol{s}$ is the vector connecting $\boldsymbol{p}(\boldsymbol{x})$ and $\boldsymbol{q}$. The dual problem of (2) can be derived via Lagrangian duality. The Lagrange dual function is given by

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) =& \inf_{\boldsymbol{s},\boldsymbol{q}} \left(\|\boldsymbol{s}\| + \lambda^\top (\boldsymbol{A}\boldsymbol{q} - \boldsymbol{b}) + \boldsymbol{\mu}^\top \left(\boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{q} - \boldsymbol{s} \right) \right) \\ =& -\boldsymbol{\lambda}^\top \boldsymbol{b} + \boldsymbol{\mu}^\top \boldsymbol{p}(\boldsymbol{x}) \\ & + \inf_{\boldsymbol{s},\boldsymbol{q}} \left(\|\boldsymbol{s}\| - \boldsymbol{\mu}^\top \boldsymbol{s} + \left(\boldsymbol{\lambda}^\top \boldsymbol{A} - \boldsymbol{\mu}^\top \right) \boldsymbol{q} \right), \end{aligned} \tag{3}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^l$, $\boldsymbol{\lambda} \succeq \mathbf{0}$ and $\boldsymbol{\mu} \in \mathbb{R}^2$ are vectors of auxiliary dual variables. For vector $\boldsymbol{s}$, the function $\|\boldsymbol{s}\| - \boldsymbol{\mu}^\top \boldsymbol{s}$ is bounded below at zero when $\|\boldsymbol{\mu}\| \leq 1$ [26, p.93], and for point $\boldsymbol{q}$, the linear function $(\boldsymbol{\lambda}^\top \boldsymbol{A} - \boldsymbol{\mu}^\top)\boldsymbol{q}$ is bounded below only when it is identically zero, i.e. $\boldsymbol{\mu} = \boldsymbol{A}^\top \boldsymbol{\lambda}$. Therefore, (3) is further simplified as

$$g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{cases} (\boldsymbol{A}\boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{b})^\top \boldsymbol{\lambda} & \|\boldsymbol{A}^\top \boldsymbol{\lambda}\| \leq 1 \\ -\infty & \text{otherwise.} \end{cases} \tag{4}$$

Then we have the dual problem of (2):

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & (\boldsymbol{A}\boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{b})^\top \boldsymbol{\lambda} \\ \text{s.t.} \quad & \left\| \boldsymbol{A}^\top \boldsymbol{\lambda} \right\| \leq 1 \\ & \boldsymbol{\lambda} \succeq \mathbf{0}. \end{aligned} \tag{5}$$

Since $\mathbb{O}$ is a convex polytope with nonempty relative interior, strong duality holds. The optimal values of problems (2) and (5) are identical and equal to the distance $\text{dist}(\boldsymbol{p}(\boldsymbol{x}), \mathbb{O})$. Consequently, the collision avoidance constraint, i.e. $\text{dist}(\boldsymbol{p}(\boldsymbol{x}), \mathbb{O}) > 0$, can now be concisely expressed using the dual vector $\boldsymbol{\lambda}$ as

$$(\boldsymbol{A}\boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{b})^\top \boldsymbol{\lambda} > 0, \quad \|\boldsymbol{A}^\top \boldsymbol{\lambda}\| \leq 1, \quad \boldsymbol{\lambda} \succeq \mathbf{0}. \tag{6}$$

B. Constraints Between an Obstacle Vertex and the Vehicle

The formulation of a 2D convex-polytope vehicle involves rotation and translation from the origin, which is expressed as

$$\mathbb{E} = \left\{ \boldsymbol{q}' \in \mathbb{R}^2 : \boldsymbol{q}' = \boldsymbol{R}(\boldsymbol{x})\boldsymbol{q} + \boldsymbol{t}(\boldsymbol{x}), \boldsymbol{G}\boldsymbol{q} \preceq \boldsymbol{g} \right\}, \tag{7}$$

where $\boldsymbol{q}$ is a point in the original convex polytope, $\boldsymbol{q}'$ is a point in the convex polytope after rotation and translation, $\boldsymbol{R}(\boldsymbol{x}) \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{t}(\boldsymbol{x}) \in \mathbb{R}^2$ respectively denote the rotation matrix and the translation vector, both calculated from the vehicle states $\boldsymbol{x}$, $\boldsymbol{G} \in \mathbb{R}^{4\times 2}$ and $\boldsymbol{g} \in \mathbb{R}^4$ contain the parameters describing a rectangle. When determining these parameters, if we place the vehicle with its rear axle center at the origin and with zero yaw angle, we have

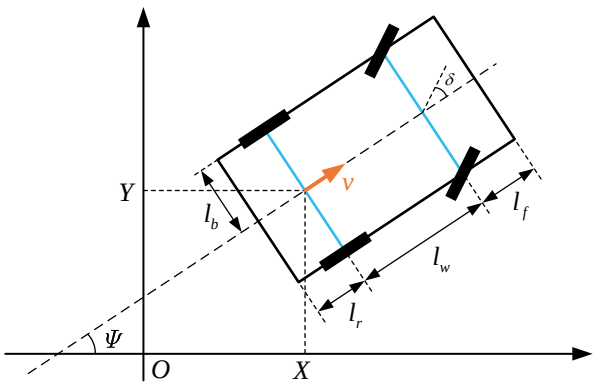

Fig. 4. Schematics of vehicle geometry and state variables.

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}^\top, \ \boldsymbol{g} = [l_w + l_f, l_b, l_r, l_b]^\top, \tag{8}$$

where l_w, l_f, l_r, and l_b, depicted in Fig. 4, correspond to the wheelbase length, the front overhang length, the rear overhang length, and half of the vehicle width, respectively. For a given obstacle vertex $\boldsymbol{p} \in \mathbb{R}^2$, the optimization problem to solve for the distance $\text{dist}(\boldsymbol{p}, \mathbb{E})$ now becomes

$$\begin{aligned} \min_{\boldsymbol{q}} \quad & \|\boldsymbol{s}\| \\ \text{s.t.} \quad & \boldsymbol{G}\boldsymbol{q} \preceq \boldsymbol{g} \\ & \boldsymbol{p} - \boldsymbol{R}(\boldsymbol{x})\boldsymbol{q} - \boldsymbol{t}(\boldsymbol{x}) = \boldsymbol{s}. \end{aligned} \tag{9}$$

Similar to Section II-A, the dual problem of (9) can be derived by analyzing the Lagrange dual function:

$$\begin{aligned} g(\boldsymbol{\lambda}, \boldsymbol{\mu}) = & -\boldsymbol{\lambda}^\top \boldsymbol{g} + \boldsymbol{\mu}^\top (\boldsymbol{p} - \boldsymbol{t}(\boldsymbol{x})) \\ & + \inf_{\boldsymbol{s},\boldsymbol{q}} \left(\|\boldsymbol{s}\| - \boldsymbol{\mu}^\top \boldsymbol{s} + \left(\boldsymbol{\lambda}^\top \boldsymbol{G} - \boldsymbol{\mu}^\top \boldsymbol{R}(\boldsymbol{x})\right) \boldsymbol{q} \right), \end{aligned} \tag{10}$$

where $\|\boldsymbol{\mu}\| \leq 1$ and $\boldsymbol{\mu} = \boldsymbol{R}^\top(\boldsymbol{x})\boldsymbol{G}^\top\boldsymbol{\lambda}$ are required for the infimum to be bounded below. Because the rotational transform preserves the Euclidean norm, the first condition becomes $\left\|\boldsymbol{G}^\top\boldsymbol{\lambda}\right\| \leq 1$. The dual problem is then written as

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \left(\boldsymbol{G}\left[\boldsymbol{R}^\top(\boldsymbol{x})(\boldsymbol{p} - \boldsymbol{t}(\boldsymbol{x}))\right] - \boldsymbol{g}\right)^\top \boldsymbol{\lambda} \\ \text{s.t.} \quad & \left\|\boldsymbol{G}^\top\boldsymbol{\lambda}\right\| \leq 1 \\ & \boldsymbol{\lambda} \succeq \boldsymbol{0}, \end{aligned} \tag{11}$$

which agrees with intuition: transform the obstacle point $\boldsymbol{p}$ inversely and perform optimization with the original convex polytope defined by $\boldsymbol{G}$ and $\boldsymbol{g}$. Here, $\boldsymbol{\lambda} \in \mathbb{R}^4, \boldsymbol{\lambda} \succeq \boldsymbol{0}$ is the dual vector. Finally, the collision avoidance constraints between an obstacle point and the vehicle are formulated as

$$\begin{aligned} \left(\boldsymbol{G}\left[\boldsymbol{R}^\top(\boldsymbol{x})(\boldsymbol{p} - \boldsymbol{t}(\boldsymbol{x}))\right] - \boldsymbol{g}\right)^\top \boldsymbol{\lambda} > 0, \\ \|\boldsymbol{G}^\top\boldsymbol{\lambda}\| \leq 1, \quad \boldsymbol{\lambda} \succeq \boldsymbol{0}. \end{aligned} \tag{12}$$

III. Problem Formulation

The cost function and other necessary constraints are described in this section to formulate an optimization problem that is compatible with the V2P constraints in Section II and the cumulative scheme in Section IV.

A. Kinematic Constraints

In parking maneuvers, vehicles are generally slow for safety concerns. With negligible wheel slip angles, the vehicle motion in low-speed (typically less than 5 m/s) scenarios can be fully captured by the kinematic bicycle model [27], which is expressed in continuous time as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Psi} \\ \dot{v} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} v\cos(\Psi) \\ v\sin(\Psi) \\ v\tan(\delta)/l_w \\ a \\ \omega \end{bmatrix}, \tag{13}$$

where $\boldsymbol{x} = [X, Y, \Psi, v, \delta]^\top \in \mathbb{R}^5$ and $\boldsymbol{u} = [a, \omega]^\top \in \mathbb{R}^2$ are vectors of the state and input variables, (X, Y), v, and a are the global position and the magnitude of velocity and acceleration of the rear axle center, Ψ is the heading, δ and ω correspond to the steering angle of the front wheel and its angular velocity, respectively. The states in vector $\boldsymbol{x}$ are illustrated in Fig. 4.

The implicit Euler method is adopted in this work for discretization. The states and inputs at time k, $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$, and the states at the next discrete time $\boldsymbol{x}_{k+1}$ are related by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h\boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k), \tag{14}$$

where h is the time step of discretization and is determined by T and N introduced in Section III-C.

B. Trust Region and Scaling

To improve solving efficiency and avoid large deviations from the warm-start positions, the method of trust region is adopted. At time k, the vehicle position is limited by a bounding box, and other vehicle states are limited by their lower and upper bounds, which are written as

$$\begin{bmatrix} X_k^{\text{WS}} - c \\ Y_k^{\text{WS}} - c \\ \underline{\Psi} \\ -\overline{v} \\ -\overline{\delta} \end{bmatrix} \preceq \begin{bmatrix} X_k \\ Y_k \\ \Psi_k \\ v_k \\ \delta_k \end{bmatrix} \preceq \begin{bmatrix} X_k^{\text{WS}} + c \\ Y_k^{\text{WS}} + c \\ \overline{\Psi} \\ \overline{v} \\ \overline{\delta} \end{bmatrix}, \tag{15}$$

where parameter c determines the size of the bounding box, $(X_k^{\text{WS}}, Y_k^{\text{WS}})$ is the position in warm-start states $\boldsymbol{x}_k^{\text{WS}}$ at time k, which is given more details in Section IV-A, the underline and overline denotes the lower and upper bounds of the corresponding variables. Note that in this work, $\underline{\Psi}$ and $\overline{\Psi}$ are set according to the headings of the start pose Ψ_S and end pose Ψ_F: $\underline{\Psi} = \min(\Psi_S, \Psi_F) - \pi$, $\overline{\Psi} = \max(\Psi_S, \Psi_F) + \pi$. The above formulation is expressed succinctly as

$$\underline{\boldsymbol{x}}(\boldsymbol{x}_k^{\text{WS}}, c) \preceq \boldsymbol{x}_k \preceq \overline{\boldsymbol{x}}(\boldsymbol{x}_k^{\text{WS}}, c). \tag{16}$$

The lower and upper bounds of the input variables can be found in Section III-E.

As scaling can be used to further accelerate convergence [28], both the states and inputs are scaled based on their value ranges. Specifically, the scale factors for states $\boldsymbol{s}_{\boldsymbol{x}} = [c, c, \pi, \overline{v}, \overline{\delta}]^\top$ and the scale factors for inputs $\boldsymbol{s}_{\boldsymbol{u}} = [\overline{a}, \overline{\omega}]^\top$.

C. Cost Function

The cost function or objective function $J \in \mathbb{R}$ is designed to minimize maneuver time T, improve passenger comfort $a^2 + v^2\omega^2$ [29], and decrease steering effort δ^2. Collision avoidance is treated as hard constraints and not incorporated into J. The cost function is written as

$$J = w_1 T + \int_{t=0}^{T} \left(w_2 \left(a^2 + v^2 \omega^2 \right) + w_3 \delta^2 \right) \mathrm{d}t, \tag{17}$$

where scalars w_1, w_2, and w_3 are corresponding weights.

The Lagrangian, i.e. the integration term, is discretized using the explicit Euler method. Therefore, the cost function can be reformulated as

$$J(T, \boldsymbol{X}, \boldsymbol{U}) = w_1 T + \sum_{k=0}^{N-1} \left(w_2 \left(a_k^2 + v_k^2 \omega_k^2 \right) + w_3 \delta_k^2 \right) h, \tag{18}$$

where $h = T/N$ is a variable derived from T, N denotes the number of input vectors, $\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N]^\top$ and $\boldsymbol{U} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{N-1}]^\top$ are the aggregation of the state and input vectors over the entire parking maneuver. The value of N is assigned at warm-start in Section IV-A.

D. Collision Avoidance Constraint Set

The collision avoidance constraint set at time k, denoted as $\mathcal{X}_k$, consists of two subsets $\mathcal{X}_k^A$ and $\mathcal{X}_k^B$, corresponding to the V2P constraints formulated in Section II-A (V2P type A) and II-B (V2P type B), respectively. The aggregation of the set and subsets over the parking maneuver are denoted as follows: $\boldsymbol{\mathcal{X}} = [\mathcal{X}_1, \ldots, \mathcal{X}_{N-1}]^\top$, $\boldsymbol{\mathcal{X}}^A = [\mathcal{X}_1^A, \ldots, \mathcal{X}_{N-1}^A]^\top$, and $\boldsymbol{\mathcal{X}}^B = [\mathcal{X}_1^B, \ldots, \mathcal{X}_{N-1}^B]^\top$. No collision avoidance constraints are imposed on the first and last discrete time.

The subsets contain different index information that is used to specify obstacles, obstacle vertices, and vehicle vertices. For V2P type A, $\mathcal{X}_k^A = \{\ldots, (i, j, m), \ldots\}$, with i for the ith obstacle in the environment, denoted as $\boldsymbol{A}_i$ and $\boldsymbol{b}_i$, m for the mth dual vector $\boldsymbol{\lambda}_m$, j and k for the jth vertex of the vehicle with states $\boldsymbol{x}_k$, denoted as $\boldsymbol{p}_j(\boldsymbol{x}_k)$ and given below:

$$\begin{aligned}
\boldsymbol{p}_0(\boldsymbol{x}_k) &= \begin{bmatrix} X_k + (l_f + l_w)\cos\Psi_k - l_b \sin\Psi_k \\ Y_k + (l_f + l_w)\sin\Psi_k + l_b \cos\Psi_k \end{bmatrix}, \\
\boldsymbol{p}_1(\boldsymbol{x}_k) &= \begin{bmatrix} X_k + (l_f + l_w)\cos\Psi_k + l_b \sin\Psi_k \\ Y_k + (l_f + l_w)\sin\Psi_k - l_b \cos\Psi_k \end{bmatrix}, \\
\boldsymbol{p}_2(\boldsymbol{x}_k) &= \begin{bmatrix} X_k - l_r\cos\Psi_k + l_b \sin\Psi_k \\ Y_k - l_r\sin\Psi_k - l_b \cos\Psi_k \end{bmatrix}, \\
\boldsymbol{p}_3(\boldsymbol{x}_k) &= \begin{bmatrix} X_k - l_r\cos\Psi_k - l_b \sin\Psi_k \\ Y_k - l_r\sin\Psi_k + l_b \cos\Psi_k \end{bmatrix},
\end{aligned} \tag{19}$$

where $j = \{0, 1, 2, 3\}$ denotes the front left, front right, rear right, and rear left vertex, respectively.

For V2P type B, $\mathcal{X}_k^B = \{\ldots, (i, j, m), \ldots\}$, with i and j for the jth vertex of the ith obstacle, denoted as $\boldsymbol{p}_{ij}$, k for $\boldsymbol{x}_k$, and m for $\boldsymbol{\lambda}_m$.

E. Optimization Problem

Navigating from the start states $\boldsymbol{x}_S$ to the end states $\boldsymbol{x}_F$, the trajectory planning problem is formulated as a nonlinear programming problem by combining (6), (12), (14), (16), and (18). The problem features high-order cost function and nonlinear constraints (nonlinearity due to the presence of trigonometric functions in kinematic constraints and the rotation matrix $\boldsymbol{R}(\boldsymbol{x}_k)$). The problem is formulated as

$$\begin{aligned}
\min_{T, \boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\Lambda}} \quad & J(T, \boldsymbol{X}, \boldsymbol{U}) \\
\text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}_S,\ \boldsymbol{x}_N = \boldsymbol{x}_F \\
& \forall\, k \in \{0, \ldots, N-1\} : \\
& \quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h\boldsymbol{f}(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k),\ \underline{\boldsymbol{u}} \preceq \boldsymbol{u}_k \preceq \overline{\boldsymbol{u}} \\
& \forall\, k \in \{1, \ldots, N-1\} : \\
& \quad \underline{\boldsymbol{x}}(\boldsymbol{x}_k^{\mathrm{WS}}, c) \preceq \boldsymbol{x}_k \preceq \overline{\boldsymbol{x}}(\boldsymbol{x}_k^{\mathrm{WS}}, c) \\
& \quad \forall\, (i, j, m) \in \mathcal{X}_k^A : \\
& \quad\quad (\boldsymbol{A}_i \boldsymbol{p}_j(\boldsymbol{x}_k) - \boldsymbol{b}_i)^\top \boldsymbol{\lambda}_m > 0, \\
& \quad\quad \|\boldsymbol{A}_i^\top \boldsymbol{\lambda}_m\| \le 1,\ \boldsymbol{\lambda}_m \succeq \boldsymbol{0} \\
& \quad \forall\, (i, j, m) \in \mathcal{X}_k^B : \\
& \quad\quad \left(\boldsymbol{G}\left[\boldsymbol{R}^\top(\boldsymbol{x}_k)(\boldsymbol{p}_{ij} - \boldsymbol{t}(\boldsymbol{x}_k))\right] - \boldsymbol{g}\right)^\top \boldsymbol{\lambda}_m > 0, \\
& \quad\quad \|\boldsymbol{G}^\top \boldsymbol{\lambda}_m\| \le 1,\ \boldsymbol{\lambda}_m \succeq \boldsymbol{0},
\end{aligned} \tag{20}$$

where constraints with > 0 are implemented as $\ge 10^{-6}$ in solvers, $\boldsymbol{\Lambda} = [\boldsymbol{\lambda}_0, \ldots, \boldsymbol{\lambda}_{M-1}]^\top$ is the aggregation of M dual vectors, $\underline{\boldsymbol{u}} = [-\overline{a}, -\overline{\omega}]^\top$ and $\overline{\boldsymbol{u}} = [\overline{a}, \overline{\omega}]^\top$ are the lower and upper bounds of the input vector, the rotation matrix $\boldsymbol{R}(\boldsymbol{x}_k)$ and the translation vector $\boldsymbol{t}(\boldsymbol{x}_k)$ are expressed as

$$\boldsymbol{R}(\boldsymbol{x}_k) = \begin{bmatrix} \cos\Psi_k & -\sin\Psi_k \\ \sin\Psi_k & \cos\Psi_k \end{bmatrix},\ \boldsymbol{t}(\boldsymbol{x}_k) = \begin{bmatrix} X_k \\ Y_k \end{bmatrix}. \tag{21}$$

IV. Trajectory Planning with Cumulative Key Constraints

A. Warm-Start Strategies

Search-based initialization using Hybrid A* [7] (HA) is adopted in this work. The resultant path is then segmented based on driving directions, i.e. straight or reverse. For each path segment, we calculate the trapezoidal velocity profile [30], which is time-optimal under acceleration and velocity bounds, and resample the path evenly in time using spline interpolation. The time interval for resampling is denoted as h_r, and N is the number of time intervals. After resampling, the initial guess for maneuver time $T^{\mathrm{HA}} = h_r N$. At discrete time k, X_k, Y_k, Ψ_k, v_k, and a_k can be initialized using the above method. The remaining δ_k and ω_k are initialized with zeros, as steering-related variables are not determinative [4]. The aggregation of the above initial guesses are denoted as $\boldsymbol{X}^{\mathrm{HA}}$ and $\boldsymbol{U}^{\mathrm{HA}}$.

The warm-start for (20) are denoted as T^{WS}, $\boldsymbol{X}^{\mathrm{WS}}$, $\boldsymbol{U}^{\mathrm{WS}}$, and $\boldsymbol{\Lambda}^{\mathrm{WS}}$. All variables except the dual get the warm-start values either from the search-based initialization or from the previous solution, denoted as T^{opt}, $\boldsymbol{X}^{\mathrm{opt}}$, and $\boldsymbol{U}^{\mathrm{opt}}$. The choice of warm-start values in each iteration is described in the next subsection.

The dual $\boldsymbol{\Lambda}$ can be warm-started by solving a second-order cone program (SOCP) originated from (5) and (11). All the dual variables are initialized only once to improve performance. In every iteration, only the newly introduced dual

variables $\boldsymbol{\Lambda}^\star$ are optimized with the current warm-start states $\boldsymbol{X}^{\text{WS}}$. The optimized dual variables are stored in a list denoted as $\boldsymbol{\Lambda}^{\text{SOCP}}$. The SOCP is given as

$$\begin{aligned}
\max_{\boldsymbol{\Lambda}^\star} \sum_{k=1}^{N-1} \sum_{\substack{(i,j,m)\\ \in \mathcal{X}_k^{A\star}}} & \left(\boldsymbol{A}_i \boldsymbol{p}_j(\boldsymbol{x}_k^{\text{ws}}) - \boldsymbol{b}_i\right)^\top \boldsymbol{\lambda}_m + \\
\sum_{k=1}^{N-1} \sum_{\substack{(i,j,m)\\ \in \mathcal{X}_k^{B\star}}} & \left(\boldsymbol{G}\left[\boldsymbol{R}^\top(\boldsymbol{x}_k^{\text{ws}})(\boldsymbol{p}_{ij} - \boldsymbol{t}(\boldsymbol{x}_k^{\text{ws}}))\right] - \boldsymbol{g}\right)^\top \boldsymbol{\lambda}_m \\
\text{s.t. } \forall\, k \in \{1, \ldots, N-1\}: & \\
\forall\, (i,j,m) \in \mathcal{X}_k^{A\star}: & \ \|\boldsymbol{A}_i^\top \boldsymbol{\lambda}_m\| \le 1,\ \boldsymbol{\lambda}_m \succeq \boldsymbol{0} \\
\forall\, (i,j,m) \in \mathcal{X}_k^{B\star}: & \ \|\boldsymbol{G}^\top \boldsymbol{\lambda}_m\| \le 1,\ \boldsymbol{\lambda}_m \succeq \boldsymbol{0},
\end{aligned} \tag{22}$$

where $\boldsymbol{\mathcal{X}}^\star$ is a new collision avoidance constraint set with its subsets at time k denoted as $\mathcal{X}_k^{A\star}$ and $\mathcal{X}_k^{B\star}$.

B. Cumulative Scheme

The method of TPCKC to solve the automated parking problem in unstructured environments is summarized in Alg. 1. Symbol $\leftarrow_k$ represents the value assignment for every possible k. A video for algorithm visualization is available in the supplementary material.

With an empty collision avoidance constraint set, the first problem is warm-started with the search-based initialization. If its solution is not collision-free, then the next problem, i.e. the first intermediate problem, is still warm-started with T^{HA}, $\boldsymbol{X}^{\text{HA}}$, and $\boldsymbol{U}^{\text{HA}}$. This is because the solution to the first problem tends to be either collision-free or severely collided, the latter being unfit for warm-start. All the other problems in the loop are warm-started with the solution to the previous problem. The loop breaks when 1) the final solution is collision-free; 2) one intermediate problem-solving fails. The second condition happens only in extremely tight cases. Adjustments to handle this condition are discussed in Section V-D.

In the loop, if the previous solution is collision-free, the problem is solved again based on that solution. This step, i.e. the final problem (trial), is necessary as we find that better warm-start and adjusted trust region may lead to improved trajectory quality. If the solution of this step is not collision-free, which rarely happens in practice if we shrink the trust region so that $c_3 < c_2$, the loop continues until another final problem (trial) is solved without collision.

If collisions are detected in the previous solution, then the corresponding constraints are enforced. The detection of collisions, i.e. violation of constraints, is simple. For V2P type A, the jth vertex of the vehicle at time k collides with the ith obstacle if $\boldsymbol{A}_i \boldsymbol{p}_j(\boldsymbol{x}_k^{\text{opt}}) \preceq \boldsymbol{b}_i$. For V2P type B, the jth vertex of the ith obstacle collides with the vehicle at time k if $\boldsymbol{G}\left[\boldsymbol{R}^\top(\boldsymbol{x}_k^{\text{opt}})(\boldsymbol{p}_{ij} - \boldsymbol{t}(\boldsymbol{x}_k^{\text{opt}}))\right] \preceq \boldsymbol{g}$. The method of k-d tree can be used for potential speedup.

We notice that if constraints are imposed only on the discrete time when a collision is detected, k for example, then in the next solution, a collision may arise at time $k+1$ or $k-1$. Therefore, the propagation of constraints is introduced in Alg. 2. All the collision avoidance constraints are propagated across time, which means that the states in a time interval before and after the collision are also constrained. The time interval is denoted as h_p, and $n_p = \text{round}(h_p/h_r)$.

Algorithm 1 Trajectory Planning with Cumulative Key Constraints (TPCKC)

1: $\mathcal{X}_k \leftarrow_k \varnothing, c \leftarrow c_1, \boldsymbol{\Lambda}^{\text{WS}} \leftarrow [\varnothing], \boldsymbol{\Lambda}^{\text{SOCP}} \leftarrow [\varnothing], \text{iter} = 0,$ $T^{\text{WS}} \leftarrow T^{\text{HA}}, \boldsymbol{X}^{\text{WS}} \leftarrow \boldsymbol{X}^{\text{HA}}, \boldsymbol{U}^{\text{WS}} \leftarrow \boldsymbol{U}^{\text{HA}}$;
2: $T^{\text{opt}}, \boldsymbol{X}^{\text{opt}}, \boldsymbol{U}^{\text{opt}} \leftarrow$ solve (20); ▷ The first problem
3: **while** True **do**
4: iter = iter + 1
5: **if** $\boldsymbol{X}^{\text{opt}}$ is not collision-free **then**
6: Collect the collision info. to a new set $\boldsymbol{\mathcal{X}}^\star$;
7: Use Alg. 2 to propagate $\boldsymbol{\mathcal{X}}^\star$;
8: **if** iter > 1 **then**
9: $T^{\text{WS}} \leftarrow T^{\text{opt}}, \boldsymbol{X}^{\text{WS}} \leftarrow \boldsymbol{X}^{\text{opt}}, \boldsymbol{U}^{\text{WS}} \leftarrow \boldsymbol{U}^{\text{opt}}$;
10: **else**
11: $T^{\text{WS}} \leftarrow T^{\text{HA}}, \boldsymbol{X}^{\text{WS}} \leftarrow \boldsymbol{X}^{\text{HA}}, \boldsymbol{U}^{\text{WS}} \leftarrow \boldsymbol{U}^{\text{HA}}$;
12: **end if**
13: Initialize $\boldsymbol{\Lambda}^\star$ with (22) and append to $\boldsymbol{\Lambda}^{\text{SOCP}}$;
14: $\mathcal{X}_k \leftarrow_k \mathcal{X}_k \cup \mathcal{X}_k^\star, c \leftarrow c_2, \boldsymbol{\Lambda}^{\text{WS}} \leftarrow \boldsymbol{\Lambda}^{\text{SOCP}}$;
15: $T^{\text{opt}}, \boldsymbol{X}^{\text{opt}}, \boldsymbol{U}^{\text{opt}} \leftarrow$ solve (20); ▷ Intermediate problems
16: **else**
17: $c \leftarrow c_3, T^{\text{WS}} \leftarrow T^{\text{opt}}, \boldsymbol{X}^{\text{WS}} \leftarrow \boldsymbol{X}^{\text{opt}}, \boldsymbol{U}^{\text{WS}} \leftarrow \boldsymbol{U}^{\text{opt}}$
18: $T^{\text{opt}}, \boldsymbol{X}^{\text{opt}}, \boldsymbol{U}^{\text{opt}} \leftarrow$ solve (20); ▷ The final problem (trial)
19: **if** $\boldsymbol{X}^{\text{opt}}$ is collision-free **then**
20: **return** $T^{\text{opt}}, \boldsymbol{X}^{\text{opt}}, \boldsymbol{U}^{\text{opt}}$; ▷ The final solution
21: **end if**
22: **end if**
23: **if** problem-solving fails in Line 15 or 18 **then**
24: **return** fatal flag;
25: **end if**
26: **end while**

Algorithm 2 Propagation of Constraints

1: Make a copy of $\boldsymbol{\mathcal{X}}$ as $\boldsymbol{\mathcal{Y}}$
2: **for** $k \in \{1, \ldots, N-1\}$ **do**
3: **for** $i \in \{-n_p, \ldots, n_p\} \setminus \{0\}$ **do**
4: **if** $1 \le k + i \le N - 1$ **then**
5: $\mathcal{Y}_k \leftarrow \mathcal{Y}_k \cup \mathcal{X}_{k+i}$
6: **end if**
7: **end for**
8: **end for**
9: Finish propagation with $\boldsymbol{\mathcal{X}} \leftarrow \boldsymbol{\mathcal{Y}}$

V. Experimental Results and Discussions

A. Simulation Setup

Simulations were implemented in Python and executed on an i9-12900KF CPU with 64 GB RAM. OpenBLAS is used with the maximum number of threads set to 1. The symbolic framework for numeric optimization CasADi [31] with the primal-dual interior point solver IPOPT [32] and the linear

TABLE I
PARAMETRIC SETTINGS ON ALL TESTED CASES

Parameter	Description	Setting
l_w	Wheelbase length	2.8 m
l_f	Front overhang length	0.96 m
l_r	Rear overhang length	0.929 m
l_b	Half of the vehicle width	0.971 m
$\overline{v}$	Upper bound of velocity	2.5 m/s
$\overline{a}$	Upper bound of acceleration	1 m/s^2
$\overline{\delta}$	Upper bound of steering angle	0.75 rad
$\overline{\omega}$	Upper bound of steering angular velocity	0.5 rad/s
w_1, w_2, w_3	Weights in the cost function	100, 5, 10
h_r	Time interval for resampling	0.04 s
h_p	Time interval for Alg. 2	0.5 s
c_1, c_2, c_3	Parameters for the trust region	1, 1.5, 1 m

solver HSL_MA97 [33] is used to solve the optimization problem (20). The modeling language for convex optimization problems CVXPY [34] with the lightweight conic solver ECOS [35] is used to solve the SOCP (22).

The benchmarks for automated parking in unstructured environments TPCAP [14] is used to validate the proposed method and perform comparisons. 18 cases are selected from a total of 20, omitting Case 7 and Case 19. The two cases are omitted for the same reason: an initialization better than HA is needed to solve them properly, which is beyond the scope of this work. Specially, Case 7 can be solved by [23] on the basis of prior knowledge. We refer the readers to the benchmark paper [14] for an overview of all cases.

The parameters used in all tested cases are listed in Table I, with the last three rows for TPCKC. All the other parameters in Table I are stipulated in the competition rules[1] of TPCAP, and we keep their values in the simulation.

B. A Close Look at Case 20

Case 20 (underground mine scenario) is chosen because it best illustrates the proposed method. The planned trajectory is exhibited in Fig. 5 and 6, and the method of TPCKC is visualized in Fig. 5. Based on the visualization, the following should be noted.

1) Non-convex obstacles are divided into convex ones.

2) The bounds on the state and input variables are met.

3) Vertex intrusion is prevented successfully by the two kinds of V2P constraints.

4) Due to resampling, the planned trajectory of TPCKC is fine-grained with $N = 396$ in Case 20. Therefore, the issue of potential violation of constraints between sampled points has been mitigated. For techniques to guarantee continuous-time trajectory safety, we refer the readers to [21] and [36].

5) We assume that the obstacle or vehicle shapes have already been dilated with a safety distance. Therefore, no further dilation is performed, and in the planned trajectory, the gaps between the vehicle and obstacles can be tiny. Objects in the visualization may overlap slightly due to the existence of line width.

6) TPCKC is able to capture key constraints and omit the irrelevant. At the time when the driving direction switches, no collision avoidance constraints are enforced even though obstacles exist in close vicinity. Also, when turning, only one side is constrained for the vehicle and obstacles.

7) Only one intermediate problem-solving is needed in Case 20, which means that the key constraints are fully captured using the first solution. In the prescribed trust region, the first solution naturally bends toward a number of obstacles, from which the collision information is collected.

8) Due to the use of maneuver time and its relatively high weight in the cost function, the velocity profile of the planned trajectory closely resembles the trapezoidal velocity profile, which is used in the resampling of HA result in Section IV-A.

[1]https://tpcap.github.io/benchmarks/

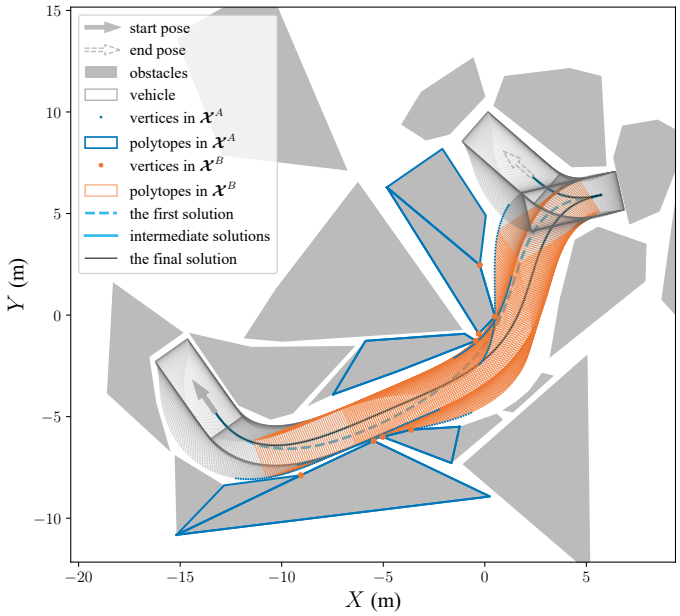

Fig. 5. Simulation result and TPCKC visualization of Case 20 (underground mine scenario).

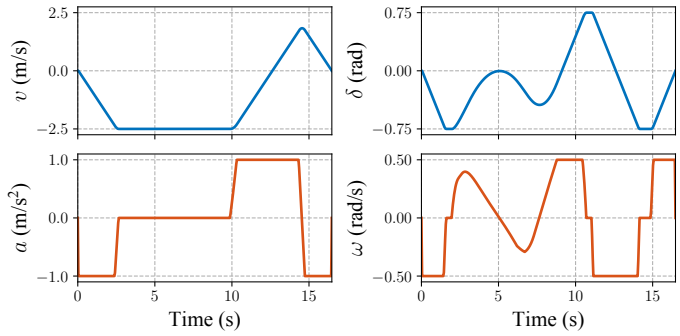

Fig. 6. States and inputs plot for the final solution of Case 20 with states in blue and inputs in red. The vehicle positions and headings, which have been visualized in Fig. 5, are not included in this plot.

C. Analysis of All Tested Cases

The visualization of planned trajectories in all tested cases is available at https://github.com/Easy121/visTPCKC. The com-

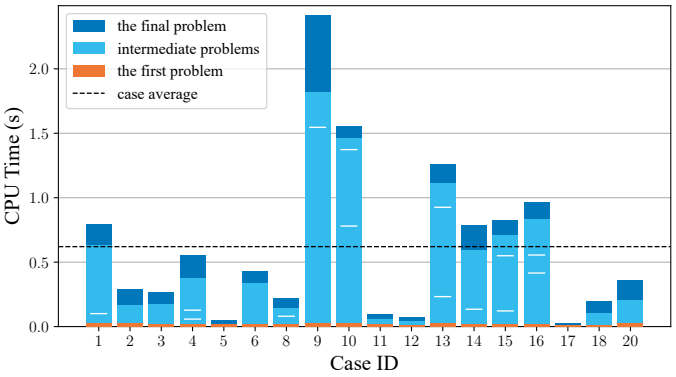

Fig. 7. CPU time of TPCKC on each tested case. If there is more than one intermediate problem, horizontal line segments in white are used for separation.

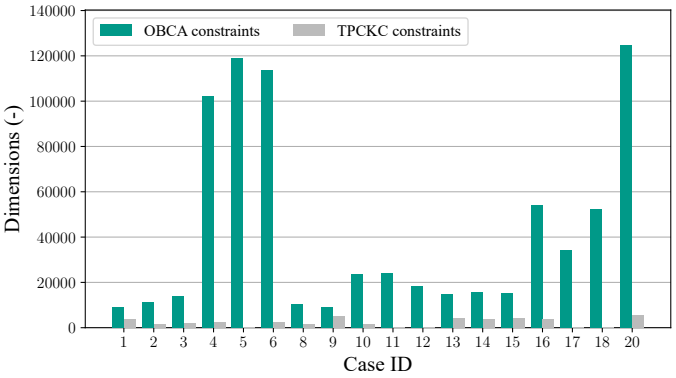

Fig. 8. Comparison of collision avoidance constraint dimensions of OBCA and TPCKC on each tested case. All pertinent equality and inequality constraints are taken into account: for OBCA, each combination of vehicle and obstacle requires $4+l+4$ dimensions of constraints, where l is the number of obstacle edges; for TPCKC, each combination of vertex and polytope requires $2+l$ (V2P type A) or $2+4$ (V2P type B) dimensions of constraints, where l is the number of polytope edges. The dimensions of TPCKC constraints are counted based on the final problem.

putational time of TPCKC, plotted in Fig. 7, is evaluated by summing the IPOPT CPU time consumed on all the problems in Alg. 1, i.e. the first problem, intermediate problems, and the final problem. The dimensions of collision avoidance constraints for OBCA (all constraints between convex sets) and TPCKC (cumulated V2P constraints) are compared in Fig. 8. From the two figures, the overall performance of TPCKC is analyzed as follows.

1) Practical real-time performance is achieved if we define it as solving with success in less than 5 s, as TPCKC completes with 0.03 s minimum, 2.41 s maximum, and 0.61 s in average.

2) The efficiency of TPCKC to capture key constraints is proved. All tested cases are solved in no more than 4 iterations, i.e. 3 times of collection of constraints, and all the final problems (trial) are solved without collision. In other words, all tested cases are solved at most 5 times (3 with collision and 2 collision-free, i.e. Cases 4, 10, 13, 15, and 16) and at least twice (the first and the final, i.e. Cases 5 and 17).

3) The first problem, i.e. trajectory planning without collision avoidance constraints in a trust region, is fast in all tested cases. Then the computational time of subsequent problems varies. The variance seems to be related to how tight the problem is, i.e. environment-sensitive. For instance, Cases 9 and 10, where both sides of the vehicle are constrained at tight bottlenecks, require the most computational time.

4) Compared to OBCA, the collision avoidance constraint dimensions of TPCKC decrease in all tested cases, which shows the effectiveness of TPCKC in reducing constraints and simplifying problems. The decrease is significant in some cases where the obstacles are more cluttered, i.e. Cases 4, 5, 6, 16, 17, 18, and 20, and relatively small in some others where the obstacles are more regularized, i.e. Cases 1, 2, 3, 8, 9, 13, 14, and 15. Note that in Cases 5 and 17, the dimensions of collision avoidance constraints are zero, which means that in these cases, the trajectory planning without collision avoidance constraints in a trust region is sufficient to generate a collision-free result.

D. Influences of Parameters

There are 5 parameters for TPCKC, 2 related to the time interval: h_r and h_p, and 3 related to the trust region: c_1, c_2, and c_3. All these parameters should be larger than 0.

The time interval for resampling, h_r, determines how fine the planned trajectory is. Smaller h_r leads to larger N, which increases the computational time but smooths the trajectory. Additionally, for extremely tight cases, decreasing h_r helps to solve successfully as vehicles need more delicate maneuvers to navigate through tight obstacles.

The time interval for the propagation of constraints, h_p, decides the influence across time of each detected collision. If h_p is infinitely large, then a collision at time k between the vehicle states $\boldsymbol{x}_k$ and an obstacle vertex $\boldsymbol{p}_{ij}$ leads to the enforcement of collision avoidance constraints between all the vehicle states $\boldsymbol{X}$ and the vertex. The decrease of h_p results in a less constrained problem for every iteration. But the risk of more rounds of collection of constraints increases: when setting $h_p = 0$, an increase in the number of iterations is witnessed in 12 cases, while Cases 5, 11, 12, 17, 18, and 20 are not influenced.

For the trust-region-related parameters, we recommend setting $c_1 < c_2$, $c_3 < c_2$, and adjusting them based on c_2. The increase of the three parameters should be treated with caution, as it may lead to severe violation of constraints or change of homotopy class. Also, c_2 should not be too small, otherwise, the problem might be infeasible.

E. Comparisons with Existing Optimization-Based Methods

The proposed method, i.e. TPCKC, is compared with three optimization-based representatives on the 18 selected cases. The methods for comparisons are H-OBCA [11], TDR-OBCA [18], and a corridor method [5]. A summary of comparisons is listed in Table II, where the three methods are abbreviated as H, T, and C, respectively. The details of computational speed and trajectory quality can be found in Table III. The computational speed is indicated by the CPU time for solving trajectory planning problems, and the trajectory quality is represented by the cost value calculated from (18) with weights specified in Table I. The visualization of all the

TABLE II
A SUMMARY OF COMPARISONS

	H	T	C	TPCKC
Time-optimal	Yes		Yes	Yes
Preserves feasible region	Yes	Yes		Yes
Iterative scheme			Yes	Yes
Success rate	83%	78%	22%	**94%**
Tops in comput. speed	4	3	0	**11**
Tops in traj. quality	9 [a]	0	0	**16**
Minimum CPU time (s) [b]	0.164	0.154	16.136	**0.027**
Maximum CPU time (s) [b]	17.931	20.566	37.659	**2.413**
Average CPU time (s) [b]	3.793	2.641	23.221	**0.611**
Average cost value (-) [b]	1840	1601	3786	**1324**

[a] 7 tops shared with TPCKC (this work).
[b] Measured on cases that are successfully solved.

tested cases solved by different methods is available at https://github.com/Easy121/visTPCKC.

The main difference between the two OBCA methods is that TDR-OBCA cancels the maneuver time from the cost function and converts the boundary conditions to cost terms for potential speedup. This makes TDR-OBCA the only method that is not time-optimal. The corridor method, as described in Section I-B, belongs to the second category that sacrifices a portion of the feasible region. Similar to TPCKC, the corridor method is iterative.

For the sake of fairness, we try our best to align the initialization strategies of the four methods and reserve differences when necessary. HA is used as the search-based initialization for H-OBCA, TDR-OBCA, and TPCKC, while an improved HA, FTHA, is used for the corridor method as they are bonded in [5]. All the search-based results are resampled using the method described in Section IV-A and the same parameter h_r. This causes the corridor method to deteriorate, as in [5], N is set to 50 for all cases and the CPU time averages 0.91 s. The choice of solver for trajectory planning is the same for all the methods (IPOPT with HSL_MA97) since it has been tested to be the best for all. For initializing the dual variables, IPOPT is used for H-OBCA and TDR-OBCA. A faster dual initialization strategy proposed along with TDR-OBCA is not used because the initialization process is not timed for comparisons. The basic cost function and corresponding weights are identical for all the methods. In TDR-OBCA and the corridor method, cost terms converted from constraints are added. In H-OBCA, TDR-OBCA, and TPCKC, constraints with > 0 are all implemented as $\geq 10^{-6}$.

From the results in Table II and III, the following can be summarized.

1) TPCKC is the fastest method under all evaluation indexes, i.e. tops in computational speed, minimum, maximum, and average CPU time.

2) Practical real-time performance is only achieved with TPCKC. All the other methods have a maximum CPU time larger than 15 s. The most time-consuming cases for the OBCA methods are Cases 6 and 20, all belonging to the cases with more cluttered obstacles. For the corridor method, Case 10 is solved for the longest time because the bottleneck is blocked by the obstacle dilation, and the vehicle makes a detour.

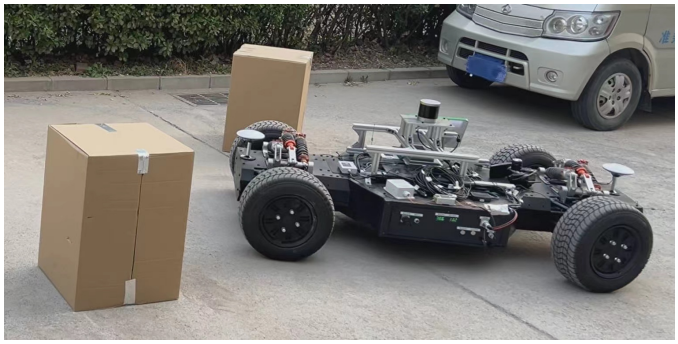

Fig. 9. Experimental platform based on a steer-by-wire and drive-by-wire chassis with two 0.5×0.5×0.7 m cardboard boxes as obstacles.

3) The speedup with TPCKC is most conspicuous in two kinds of cases: cases where the obstacles are more cluttered, e.g. Case 20 with 97.4% CPU time reduced, and cases where the trajectory planning in a trust region with few or no collision avoidance constraints leads to a collision-free result, e.g. Case 5 with 97.1% CPU time reduced and Case 11 with 77.8% reduced.

4) TPCKC has the highest success rate. As TPCKC neglects the rare case of overlapping of convex polytopes without vertex intrusion, it fails in Case 14 with a needle-like obstacle. Such obstacles are challenging for the other methods as well, with H-OBCA failing in Cases 13 and 15, and TDR-OBCA in Cases 14 and 15. In Case 18, H-OBCA fails to find a trajectory that satisfies all the constraints, and the infeasibility flag is returned. All the tested cases are solved without a fatal flag by TDR-OBCA. However, deviations from the end states are found in Cases 1 and 18, resulting from the relaxation of the boundary conditions. The failures of the corridor method are mainly due to the obstacle dilation.

5) TPCKC has the highest trajectory quality, with 16 tops and the smallest cost value on average. One reason is that TPCKC involves rounds of warm-start with increasing quality, which helps find better solutions than the methods only warm-started once with search-based initialization. Also, the nonlinearity is reduced due to fewer constraints, and better local minima can be found. In 9 cases, TPCKC manages to find better solutions, and in 7 cases, the solutions of TPCKC and H-OBCA agree with each other. Nevertheless, it is still possible for TPCKC, a nonlinear method, to converge to poor local minima, e.g. Case 4.

F. Real-World Experiment

We have also validated the ability of TPCKC to generate collision-free and easy-to-follow trajectories in a real-world experiment. The experimental platform is shown in Fig. 9. Two irregularly placed and moveable cardboard boxes are used as obstacles to construct a general unstructured scenario with possible environmental changes. The box shape also helps to ease the burden of perception and guarantee accurate detection from all directions. The parameters that are changed for the experiment are updated in Table IV. Note that the maximum

TABLE III
COMPARISONS OF COMPUTATIONAL SPEED AND TRAJECTORY QUALITY

Case ID		1	2	3	4	5	6	8	9	10
CPU Time (s)	H-OBCA	**0.2805**	**0.1819**	1.4144	10.3260	1.9501	17.9309	0.2322	1.9165	**0.5272**
	TDR-OBCA	– [c]	0.2007	0.4414	3.9377	1.5794	2.1609	**0.1539**	**0.3279**	0.9246
	Corridor	– [d]	– [d]	– [d]	– [d]	– [d]	– [d]	– [d]	– [d]	37.6589
	TPCKC (this work)	0.7957	0.2917	**0.2688**	**0.5522**	**0.0465**	**0.4283**	0.2206	2.4128	1.5544
Cost Value (-)	H-OBCA	1413.25	1447.77	**1335.49**	**1212.36**	802.62	1340.39	**1261.22**	9079.29	**1451.93**
	TDR-OBCA	– [c]	1682.01	1630.17	1322.69	965.06	1652.84	1546.17	2041.47	1985.26
	Corridor	– [d]	– [d]	– [d]	– [d]	– [d]	– [d]	– [d]	– [d]	6234.04
	TPCKC (this work)	**1269.33**	**1391.92**	**1335.49**	1421.33	**802.57**	**1340.04**	**1261.22**	**1820.56**	**1451.93**
Case ID		11	12	13	14	15	16	17	18	20
CPU Time (s)	H-OBCA	1.0464	0.1643	– [a]	**0.2373**	– [a]	3.9636	2.8767	– [b]	13.8490
	TDR-OBCA	0.4539	0.5889	**0.5474**	– [a]	– [a]	4.6281	0.4649	– [c]	20.5658
	Corridor	22.0575	16.1358	– [d]	– [d]	– [d]	17.0309	– [d]	– [d]	– [d]
	TPCKC (this work)	**0.1009**	**0.0705**	1.2618	– [e]	**0.8245**	**0.9668**	**0.0273**	**0.1989**	**0.3609**
Cost Value (-)	H-OBCA	**1545.89**	**1223.21**	– [a]	**1390.52**	– [a]	1648.48	**696.11**	– [b]	**1754.15**
	TDR-OBCA	1949.85	1546.81	1563.78	– [a]	– [a]	1608.28	785.61	– [c]	2130.67
	Corridor	2078.04	1601.87	– [d]	– [d]	– [d]	5230.59	– [d]	– [d]	– [d]
	TPCKC (this work)	**1545.89**	**1223.21**	**1460.85**	– [e]	**1316.26**	**1539.21**	**696.11**	**879.19**	**1754.15**

[a] Solves with success but vertex intrusion is detected with a needle-like obstacle.
[b] Terminates due to Infeasible_Problem_Detected error.
[c] Solves with success but converges to a local minimum where $\boldsymbol{x}_N$ deviates from $\boldsymbol{x}_F$.
[d] Search fails because $\boldsymbol{x}_S$ or $\boldsymbol{x}_F$ is inside the obstacles dilated with the two-disk approximation.
[e] Solves with success but overlapping between the vehicle and a needle-like obstacle without vertex intrusion is detected.

TABLE IV
UPDATED PARAMETRIC SETTINGS FOR THE REAL-WORLD EXPERIMENT

Parameter	Setting	Parameter	Setting
l_w	1.474 m	$\overline{v}$	0.6 m/s
l_f	0.32 m	$\overline{a}$	0.45 m/s^2
l_r	0.225 m	$\overline{\delta}$	0.32 rad
l_b	0.56 m	$\overline{\omega}$	0.45 rad/s

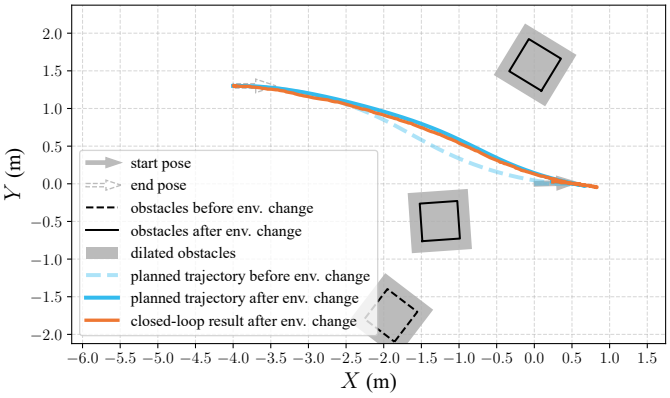

Fig. 10. Planned trajectory before and after the environmental change and closed-loop trajectory tracking performance. The obstacles are dilated with a distance of 0.15 m.

steering angle of the chassis is smaller than the one used in the simulation (Table I). For localization, the chassis uses an accurate global positioning system integrating GNSS and motion sensors. For perception, with the point clouds from a 32-channel lidar, the boxes are detected as obstacles using ground segmentation [37], point cloud clustering [38], and L-shape fitting [39]. The trajectory planning is performed on a Jetson AGX Xavier with Carmel CPU.

In the experiment, once the chassis starts following a planned trajectory, one obstacle is moved near. When a future collision is detected, the chassis stops, replans, and follows a new trajectory. The planned trajectory is tracked using an offset-free nonlinear model predictive controller, which will be introduced in our future work. The results of the experiment are plotted in Fig. 10. The reliability of TPCKC under environmental changes and its ability to generate easy-to-follow trajectories are shown. A video of the experiment and the corresponding ROS [40] visualization is available in the supplementary material.

VI. CONCLUSIONS

In this work, we formulate two kinds of vertex-to-polytope (V2P) constraints and propose the method of trajectory planning with cumulative key constraints (TPCKC) to simplify the trajectory planning problem while preserving the feasible region. The V2P constraints serve to target vertex intrusion and make it possible to omit irrelevant constraints on trivial vertices. The method of TPCKC builds on the V2P constraints and serves to capture key constraints systematically and cumulatively. The efficiency and effectiveness of the proposed method are validated based on 18 cases selected from TPCAP, benchmarks for automated parking in unstructured environments. With the nonlinearity reduced, the proposed method excels in both computational speed and trajectory

quality when compared with three optimization-based representatives. Practical real-time performance on all tested cases, together with the highest success rate and trajectory quality, is achieved. Additionally, a real-world experiment that validates the proposed method is conducted.

Limitations and future work: While the simulations and experiment show the superior performance and validated reliability of TPCKC in unstructured environments, environment-sensitive variance in computational time and the problem of poor local minima still arise and should be dealt with. Furthermore, trajectory planning confronted with perception uncertainties and dynamic environments are two important future directions, which may require joint research efforts in perception and decision-making systems.

REFERENCES

[1] 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 Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[2] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[3] W. Liu, Z. Li, L. Li, and F.-Y. Wang, "Parking Like a Human: A Direct Trajectory Planning Solution," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3388–3397, Dec. 2017.

[4] Y. Guo, D. Yao, B. Li, H. Gao, and L. Li, "Down-Sized Initialization for Optimization-Based Unstructured Trajectory Planning by Only Optimizing Critical Variables," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 709–720, Jan. 2023.

[5] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-Based Trajectory Planning for Autonomous Parking With Irregularly Placed Obstacles: A Lightweight Iterative Framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 970–11 981, Aug. 2022.

[6] S. Manzinger, C. Pek, and M. Althoff, "Using Reachable Sets for Trajectory Planning of Automated Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 232–248, Jun. 2021.

[7] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, Apr. 2010.

[8] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning." in *Robotics: Science and Systems*, 2007, pp. 326–333.

[9] S. M. LaValle, *Planning Algorithms*. Cambridge university press, 2006.

[10] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined Speed and Steering Control in High-Speed Autonomous Ground Vehicles for Obstacle Avoidance Using Model Predictive Control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, Oct. 2017.

[11] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous Parking Using Optimization-Based Collision Avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*, Dec. 2018, pp. 4327–4332.

[12] Z. Li, P. Zhao, C. Jiang, W. Huang, and H. Liang, "A Learning-Based Model Predictive Trajectory Planning Controller for Automated Driving in Unstructured Dynamic Environments," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 5944–5959, Jun. 2022.

[13] X. Zhang, W. Zhang, Y. Zhao, H. Wang, F. Lin, and Y. Cai, "Personalized Motion Planning and Tracking Control for Autonomous Vehicles Obstacle Avoidance," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4733–4747, May 2022.

[14] B. Li, L. Fan, Y. Ouyang, S. Tang, X. Wang, D. Cao, and F.-Y. Wang, "Online Competition of Trajectory Planning for Automated Parking: Benchmarks, Achievements, Learned Lessons, and Future Perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 16–21, Jan. 2023.

[15] M. Vitus, V. Pradeep, G. Hoffmann, S. Waslander, and C. Tomlin, "Tunnel-milp: Path planning with sequential convex polytopes," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, p. 7132.

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

[17] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, May 2021.

[18] R. He, J. Zhou, S. Jiang, Y. Wang, J. Tao, S. Song, J. Hu, J. Miao, and Q. Luo, "TDR-OBCA: A Reliable Planner for Autonomous Driving in Free-Space Environment," in *2021 American Control Conference (ACC)*, May 2021, pp. 2927–2934.

[19] K. Bergman, O. Ljungqvist, and D. Axehill, "Improved Path Planning by Tightly Combining Lattice-Based Path Planning and Optimal Control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 57–66, Mar. 2021.

[20] M. Wang, L. Zhang, Z. Zhang, and Z. Wang, "A Hybrid Trajectory Planning Strategy for Intelligent Vehicles in On-Road Dynamic Scenarios," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 2832–2847, Mar. 2023.

[21] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014.

[22] C. Liu, C.-Y. Lin, and M. Tomizuka, "The Convex Feasible Set Algorithm for Real Time Optimization in Motion Planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, Jan. 2018.

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

[24] D. Zermas, I. Izzat, and N. Papanikolopoulos, "Fast segmentation of 3D point clouds: A paradigm on LiDAR data for autonomous vehicle applications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5067–5073.

[25] J. M. Keil, "Decomposing a polygon into simpler components," *SIAM Journal on Computing*, vol. 14, no. 4, pp. 799–817, 1985.

[26] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge university press, 2004.

[27] R. Rajamani, *Vehicle Dynamics and Control*. Springer Science & Business Media, Dec. 2011.

[28] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. SIAM, 2010.

[29] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "An Admissible Heuristic to Improve Convergence in Kinodynamic Planners Using Motion Primitives," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 175–180, Jan. 2020.

[30] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 3248–3253.

[31] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[32] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[33] "Coin-HSL: A collection of HSL packages for use with IPOPT," https://licences.stfc.ac.uk/product/coin-hsl.

[34] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[35] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *European Control Conference (ECC)*, 2013, pp. 3071–3076.

[36] M. d. S. Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, "Collision-Free Encoding for Chance-Constrained Nonconvex Path Planning," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, Apr. 2019.

[37] H. Lim, M. Oh, and H. Myung, "Patchwork: Concentric Zone-Based Region-Wise Ground Segmentation With Ground Likelihood Estimation Using a 3D LiDAR Sensor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6458–6465, Oct. 2021.

[38] Y. Li, C. Le Bihan, T. Pourtau, T. Ristorcelli, and J. Ibanez-Guzman, "Coarse-to-Fine Segmentation on LiDAR Point Clouds in Spherical Coordinate and Beyond," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14 588–14 601, Dec. 2020.

[39] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient L-shape fitting for vehicle detection using laser scanners," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 54–59.

[40] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system,” May 2018.

Zijun Guo received the B.S. degree in 2022 from the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China, where he is currently working toward the Ph.D. degree. In the IEEE ITSC 2022, he won first prize in the automated parking competition TPCAP. His current research interests include vehicle control, trajectory planning, decision making, and machine learning.

Yuanxin Wang received the B.E. degree from the Beijing Institute of Technology (BIT), Beijing, China in 2024, where he is currently working toward the M.E. degree. His current research interests include vehicle control and trajectory planning.

Huilong Yu (Member, IEEE) received the M.Sc. degree in mechanical engineering from Beijing Institute of Technology, and the Ph.D. degree in mechanical engineering from Politecnico di Milano, Milano, Italy, in 2013 and 2017, respectively. He was a postdoctoral Research Fellow of advanced vehicle engineering with the University of Waterloo, Waterloo, CA, from 2018 to 2021. He is now a Professor with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China. His research interests include vehicle dynamics, vehicular optimal design and control, and autonomous driving.

Junqiang Xi (Member, IEEE) received the B.S. degree in automotive engineering from the Harbin Institute of Technology, Harbin, China, in 1995, and the Ph.D. degree in vehicle engineering from the Beijing Institute of Technology (BIT), Beijing, China, in 2001. In 2001, he joined the State Key Laboratory of Vehicle Transmission, BIT. During 2012-2013, he made research as an Advanced Research Scholar in Vehicle Dynamic and Control Laboratory, Ohio State University, Columbus, OH, USA. He is currently a Professor and Director of the Automotive Research Center in BIT. His research interests include vehicle dynamics and control, powertrain control, mechanics, intelligent transportation systems, and intelligent vehicles.