Trajectory Planning for an Autonomous Vehicle in Spatially Constrained Environments

Yuqing Guo, Danya Yao, *Member, IEEE*, Bai Li, *Member, IEEE*, Zimin He, Haichuan Gao, and Li Li, *Fellow, IEEE*

***Abstract*— Road shoulders and slopes often appear in unstructured environments. They make 2.5D vehicle trajectory planning commonly seen in our daily life, which lies on a 2D manifold embedded in a 3D space. The height difference of these terrains brings spatially dependent constraints on vehicle maneuvers, such as the limit on vehicle steering for vehicle tire protection when a vehicle approaches a road shoulder edge. These constraints have an "if-else" structure since they are activated only when the vehicle passes through the local area with a height difference, making the restriction on variables coupled with the judgment of variables. This makes the application of state-of-art optimization-based planners challenging. To solve this problem, we devise an approximation formulation for these constraints in the trajectory planning optimization problem, whose solution depends on a proper initial guess for the optimizer. We propose a two-stage trajectory planning framework, where the first stage improves the hybrid A* algorithm by adding spatially dependent constraints into node expansion to provide the initial guess. Then, the optimization problem with the formulated spatially dependent constraints is solved for further trajectory smoothness and quality. Finally, the simulation results validate the fast and high-quality planning performance of our proposed framework.**

***Index Terms*— Autonomous driving, trajectory planning, spatially dependent constraints.**

I. Introduction

AUTONOMOUS vehicles are expected to significantly modify travel behaviors [1], [2]. In daily life, autonomous vehicles often have to solve trajectory planning problems with a height difference, such as passing through road shoulders or gentle slopes. The planning lies on a 2D manifold embedded in a 3D space, which is 2.5D planning [3]. Compared with 2.5D off-road terrains, the flat terrains in this work can be regarded as a piecewise linear hyperplane [4], which can eliminate the need to design all steering wheel maneuvers and convert 2.5D planning into 2D planning to reduce the computational complexity associated with the increased planning dimension [5], [6].

However, the height difference brings extra spatially dependent constraints to the 2D planning problem. Let us take Fig. 1 for an example. When an autonomous vehicle wants to park on the road shoulder due to the narrow parking space, there is a constraint on vehicle orientation, which requires the vehicle to be as perpendicular as possible to the road shoulder when the vehicle is approaching the edge of the shoulder. The constraint has an "if-else" structure, i.e., if the vehicle is in the specific area with a height difference, it will be activated, otherwise, it will not. Ignoring the constraint will cause serious damages to the vehicle, such as blowout and tire cord breakdown [7].

The "if-else" constraint plus characteristics of our concerned problem make planning challenging. Here are as follows. 1) Areas with spatially dependent terrain constraints (like the edge of the shoulder) may be very small compared with the whole planning configuration space, which makes it difficult to predetermine the waypoints in the small local area with a height difference. 2) There are various static and dynamic obstacles in the environment, such as pedestrians and pets, the behaviors of which are difficult to predict. 3) There is not a

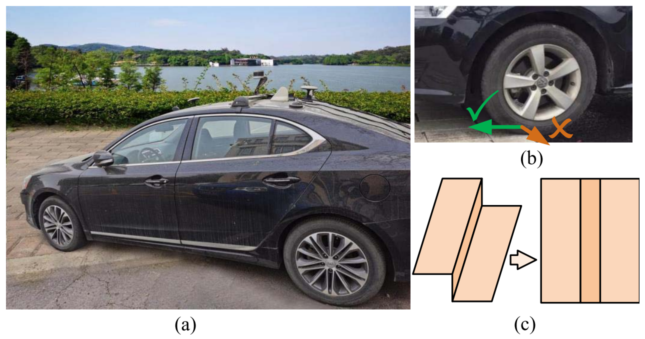


Fig. 1. An illustration of the spatially dependent constraint on the vehicle orientation. (a) An autonomous vehicle aims to traverse a road shoulder on its way to a desired parking spot; (b) A blowout is likely to occur if the included angle between the steering wheels and the shoulder curb is overly small; (c) The planning plane is a piecewise linear hyperplane and can be tiled to a 2D plane.

Manuscript received 22 April 2021; revised 14 November 2021 and 17 February 2022; accepted 30 March 2022. Date of publication 6 April 2022; date of current version 11 October 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2020AAA0108104; in part by the National Natural Science Foundation of China under Grant 62133002 and Grant 62103139; in part by the Fundamental Research Funds for the Central Universities under Grant 531118010509; and in part by the Natural Science Foundation of Hunan Province, China, under Grant 2021JJ40114. The Associate Editor for this article was J. W. Choi. *(Corresponding author: Li Li.)*

Yuqing Guo, Zimin He, and Haichuan Gao are with the Department of Automation, Tsinghua University, Beijing 100084, China.

Danya Yao is with the Department of Automation, BNRist, Tsinghua University, Beijing 100084, China, and also with the Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, Nanjing 210096, China (e-mail: yaody@tsinghua.edu.cn).

Bai Li is with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha 410082, China (e-mail: libai@zju.edu.cn).

Li Li is with the Department of Automation, BNRist, Tsinghua University, Beijing 100084, China (e-mail: li-li@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TITS.2022.3164548

1558-0016 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

nominal reference line with which the planning problem can be described in an easier coordinate system.

Searching-based planners, such as Dijkstra and A* [8], or sampling-based planners, such as RRT [9], can cope with the above difficulties for holonomic robots [10], [11]. Ganganath *et al.* utilized an A*-like heuristic search algorithm for a mobile robot on non-flat surfaces [12] and Wang *et al.* planned trajectories with RRT for a robotic wheelchair in a sloped environment [3]. However, compared to these holonomic robots, an autonomous vehicle has more complex kinematic constraints and harsher requirements w.r.t. trajectory smoothness, thereby making the aforementioned algorithms inapplicable [13].

Recently, the combination of searching and optimization is emerging as an effective approach to obtain a smooth trajectory for autonomous vehicles [14]. It refers to finding a coarse trajectory quickly with a searching-based planner first, then describing the planning task as an optimal control problem (OCP) and using a gradient-based optimizer to solve it [15]. It provides theoretical guarantees and stable high-quality solutions compared with intelligent optimization algorithms, such as Particle Swarm Optimization (PSO) [16] or deep learning methods [17], [18], whose instability may make them fail to find a safe feasible trajectory in applications. This combination acquires the searching-based methods' high solution speed ability and the optimization-based methods' high quality, making it suitable for autonomous driving [19], [20]. However, gradient-based optimizers typically cannot deal with constraints with "if-else" conditions because it is difficult to compute their gradients.

There are two ways to handle the "if-else" constraints. One way is to set a series of subgoals based on different terrain features, and then decompose the whole planning problem into a series of subproblems with different constraints, which are widely applied for mountain slopes [5], [6]. However, compared with mountain slopes, our local area with spatially dependent constraints occupies a very small proportion in the entire planning configuration space, and may only involve a few waypoints. Subregional trajectory planning will greatly limit the trajectory shape in this area, making it hard to change the whole trajectory significantly for further smoothness. The other way is to optimize the time points at the sampling spatial positions of a predefined reference line to deal with spatially dependent constraints [21]. This type of method is efficient for scenarios with a reference line but becomes inapplicable in dealing with our concerned problem. Overall, the above methods cannot handle our concerned problem.

To address the limitations of the above methods, this paper proposes a two-stage trajectory planning framework for 2.5D planning. Considering the core difficulty stems from the gradient-based optimizer's inability to handle constraints coupled with judgments, we design an approximation formulation for these constraints in the OCP. We construct two expressions on decision variables judgment and restriction, and require their multiplication equal to zero on all waypoints. The expression of the judgment is not equal to zero only when the vehicle is in a specific area, so the restriction is activated only at this time. Although it complicates optimization computation for imposing constraints on all waypoints rather than some fixed points, it can ensure safety and adjust the whole trajectory for enough quality and smoothness. The formulation increases the OCP non-convexity, which requires a proper initial guess to accelerate the convergence to an optimum. Therefore, the first stage improves the hybrid A* algorithm by adding spatially dependent constraints to provide such an initial guess for the second stage optimizer.

The contributions of this paper are summarized as follows. 1) This paper proposes a two-stage trajectory planning framework for 2.5D planning with spatially dependent constraints. 2) Based on our proposed spatially dependent constraint handling method, we formulate the planning optimization problem with other complex constraints, which can be efficiently solved with gradient-based optimizers. 3) We illustrate how the proposed trajectory planning framework can be applied to several challenging cases.

The rest of this paper is organized as follows. Section II presents the coordinate transformation from 2.5D to 2D planning and the formulation of the trajectory planning optimization problem. Section III introduces the spatially dependent constraint handling method and the two-stage trajectory planning framework. The effectiveness of the proposed framework is validated via simulations in Section IV. Section V concludes this paper briefly.

II. 2.5D Problem Formulation

This section first transfers 2.5D trajectory planning into 2D trajectory planning by coordinate conversion and then formulates it as an optimization problem.

A. Coordinate Transformation

We transform 2.5D planning into 2D planning by unfolding the hyperplane to reduce computational complexity since the 2.5D traveling plane can be seen as a piecewise linear hyperplane [4]. Fig. 2 shows the schematic of the coordinate transformation, where Plane I and Plane II are joined by a sloping plane. Plane I is defined as the xoy plane and the direction perpendicular to it is defined as the z-axis of the 2.5D world coordinate system $W_{2.5}$. The slope inclination is denoted by α and the road shoulder can be seen as a slope with $\alpha = \pi/2$. The slope height is denoted by h.

The unfolding plane of the traveling surface is defined as the xoy plane in the 2D world coordinate system W_2. The width of Plane I is y_a and the width of Slope O is $h/\sin\alpha$, so the ordinate y_b for the top of Slope O is $y_a + h/\sin\alpha$. The width of Plane II is e, so the ordinate for the top of Plane II in the 2D coordinate system y_c is $y_b + e$. We set I_2 as the coordinate of a point on the 2D plane, $I_{2.5}$ as the 2.5D coordinate of the point, and E as the conversion matrix. Therefore, the coordinate conversion relationship is

$$I_2 = E I_{2.5}. \tag{1}$$

B. 2.5D Trajectory Planning Problem Formulation

The trajectory planning problem is formulated as an optimization problem with decision variables, cost functions, and constraints.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

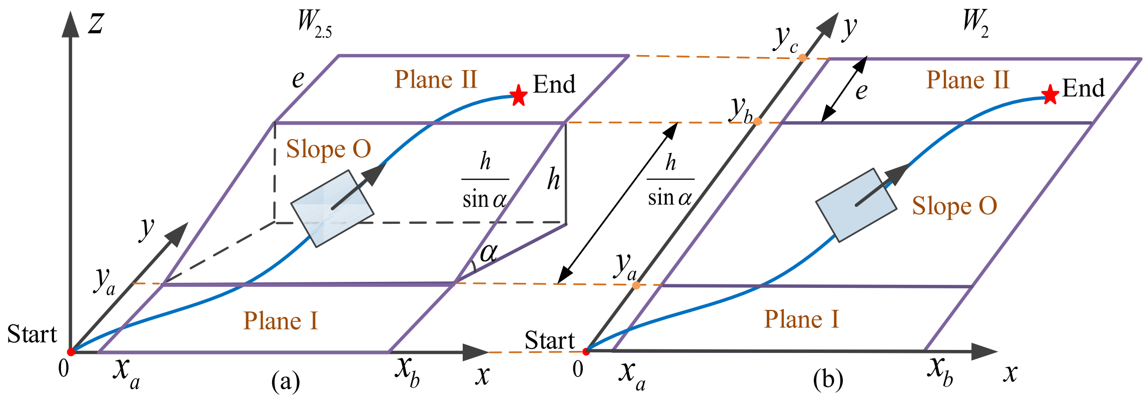


Fig. 2. A schematic of coordinate transformation by unfolding the terrain surface. (a) The 2.5D coordinate system; (b) The transformed 2D coordinate system.

1) Decision Variables: The trajectory $\boldsymbol{x}$ can be described by a series of waypoints $\boldsymbol{x} = \left[\boldsymbol{x}(0)^{\mathrm{T}}, \boldsymbol{x}(1)^{\mathrm{T}}, \ldots, \boldsymbol{x}(n)^{\mathrm{T}}, t_n\right]^{\mathrm{T}}$, where the step size of the trajectory is n. $\boldsymbol{x}(k)$ denotes the state of the vehicle at the time step $k\Delta t$, where $\Delta t = t_n/n$ and t_n is the whole traveling time. Since the concerned trajectory planning scheme is conducted in a low-speed scenario, and the vehicle wheels always maintain good contact with the ground, an autonomous vehicle can be assumed as a rigid body without considering coupled dynamics of vehicle suspensions, tire sideslips, and all-steering maneuvers. So we adopt the bicycle model [22] to describe the vehicle kinematics, as shown in Fig. 3.

The length and width of the vehicle are denoted by L and W, respectively. $\mathrm{L_m}$ is the wheelbase. $P(k) = [p_x(k), p_y(k)]^{\mathrm{T}}$ refers to the mid-point of the rear wheel axis. $\theta(k)$, $v(k)$, and $\varphi(k)$ represent the vehicle orientation, velocity, and steering angle of the vehicle, respectively. Particularly, the front-wheel steering angle $\varphi(k)$ is defined as the angle between the wheel orientation and the vehicle orientation. The angle $\tau(k)$ refers to the angle between the front wheels and the horizontal direction, which is defined as follows:

$$\tau(k) = \begin{cases} \theta(k) + \varphi(k), & -\dfrac{\pi}{2} \leq \theta(k) \leq \dfrac{\pi}{2} \\ \theta(k) - \varphi(k), & \text{otherwise.} \end{cases} \tag{2}$$

We use $\boldsymbol{x}(k) = [p_x(k), p_y(k), \theta(k), v(k), \varphi(k)]^{\mathrm{T}}$ to represent the state of the vehicle at $k\Delta t$. The dimension of the decision variable $\boldsymbol{x} = [\boldsymbol{x}(0)^{\mathrm{T}}, \boldsymbol{x}(1)^{\mathrm{T}}, \ldots, \boldsymbol{x}(n)^{\mathrm{T}}, t_n]^{\mathrm{T}}$ is $5(n+1)+1$.

2) Cost Function: For low-speed trajectory planning schemes, the to-be-optimized cost function is usually set to achieve a trajectory with the minimum distance, traveling time [22], or jerk variation [23]. In this work, it is defined as

$$J(\boldsymbol{x}) = \omega_1 t_n + \omega_2 \sum_{k=1}^{n-1} \|v(k+1) - v(k)\|_2 + \omega_3 \sum_{k=1}^{n-1} \|\varphi(k+1) - \varphi(k)\|_2 \tag{3}$$

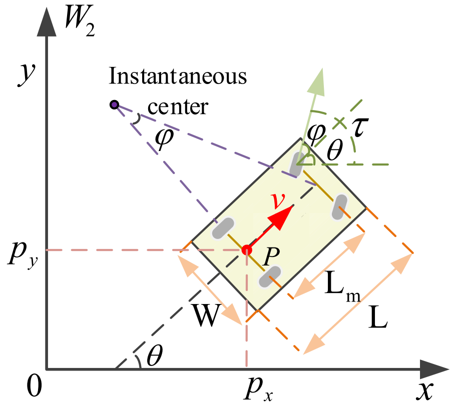


Fig. 3. A schematic of the vehicle kinematics related parameters in using the bicycle model. For the sake of simplicity, we omit the time representation k.

to achieve a trade-off between traveling time and trajectory smoothness. Here, the weights ω_1, ω_2, and ω_3 are set as 0.5, 0.2, and 0.3, respectively.

3) Constraints: In this subsection, the constraints in trajectory planning for autonomous vehicles are formulated, which include state constraints, kinematic constraints, collision avoidance constraints, and spatially dependent constraints.

a) Initial and terminal state constraints: The initial and terminal constraints are mainly limited by the state of the vehicle at the initial and terminal time steps:

$$\boldsymbol{x}(0) = \boldsymbol{x}_{\text{start}}, \quad \boldsymbol{x}(n) = \boldsymbol{x}_{\text{end}}, \tag{4}$$

where $\boldsymbol{x}_{\text{start}}$ and $\boldsymbol{x}_{\text{end}}$ are the initial and terminal states.

b) Kinematic inequality constraints: An autonomous vehicle has to satisfy the limits of physical restrictions on the values of velocity, acceleration, and steering angle within their corresponding bounds:

$$v_{\min} \leq v(k) \leq v_{\max}, \tag{5}$$
$$a_{\min} \leq a(k) \leq a_{\max}, \tag{6}$$
$$\varphi_{\min} \leq \varphi(k) \leq \varphi_{\max}. \tag{7}$$

c) Kinematic equality constraints: According to the bicycle model [22], [24], an autonomous vehicle should satisfy the

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

kinematic equality constraints:

$$p_x(k+1) - p_x(k) = \frac{t_n}{n}\upsilon(k)\cdot\cos\theta(k), \tag{8}$$

$$p_y(k+1) - p_y(k) = \frac{t_n}{n}\upsilon(k)\cdot\sin\theta(k), \tag{9}$$

$$\theta(k+1) - \theta(k) = \frac{t_n}{n}\upsilon(k)\cdot\frac{\tan\varphi(k)}{\mathrm{L_m}}. \tag{10}$$

d) Collision avoidance constraints: An autonomous vehicle needs to avoid collisions with all dynamic and static obstacles. Since our focus is the planning layer of autonomous vehicles, we assume that the perception and prediction layers can well perceive and predict the space occupied by obstacles at each time step [25]. It should be mentioned, the time step Δt is fixed and t_n changes with n, when dealing with dynamic obstacles.

The obstacle space is defined as $\mathcal{X}_{\text{obs}}$ and collision-free state space is constrained as

$$m(\boldsymbol{x}) \in \mathcal{X}_{\text{free}} = \mathcal{X}\backslash\mathcal{X}_{\text{obs}}, \tag{11}$$

where function $m(\cdot)$ is the mapping function from vehicle state to vehicle body.

The above formulation can guarantee the safety of discrete waypoints, but cannot strictly guarantee that the connecting lines between adjacent waypoints are also collision-free. To ensure the trajectory feasibility, the waypoints need to be very dense, making the whole trajectory is nearly collision-free. Moreover, to guarantee strict safety, after obtaining discrete waypoints, we use [26] to further check collisions along the whole continuous trajectory and use [27] to wrap the local trajectory to go around obstacles if the connecting line between waypoints intersects with obstacles.

e) Spatially dependent constraints: The spatially dependent constraints require that they are activated only when the ego vehicle enters a specific area. Different from the above constraints, the restriction of spatially dependent constraints on variables is coupled with the judgment of variables, making it difficult to handle. In Section III.A, we will describe the constraint handling method in detail.

III. Proposed Trajectory Planner

This section introduces the two-stage trajectory planning framework, where we first introduce the spatially dependent constraint handling method in the OCP. Although the proposed strategy can guarantee safety, it increases the OCP non-convexity. To accelerate the convergence to an optimum, the first stage provides a high-quality initial guess for the second stage optimizer by searching for a coarse trajectory satisfying spatially dependent constraints with our improved hybrid A* algorithm. Then, the second stage adopts a gradient-based method to solve the OCP for enough trajectory quality and feasibility. The whole flow chart is shown in Alg. 1.

A. Spatially Dependent Constraint Handling

The spatially dependent constraints are activated only when the ego vehicle enters a specific area. For this purpose, we convert the spatially dependent constraints into two requirements:

Algorithm 1 Proposed 2.5D Two-Stage Planner

```
Input: Obstacle information, the ego vehicle information, and
    environment information;
Output: An optimal trajectory;
1: Initialization;
2: $r \leftarrow 0$, $\lambda_0 \leftarrow 10^4$;
3: $\sum F_{\max}(\boldsymbol{x}) \leftarrow +\infty$;
4: $2D \leftarrow$ TransformCoordinate(2.5D);
5: First stage: trajectory searching
6: Dilated obstacles $\leftarrow$ DilateObstacle(obstacle);
7: $\{p_x^0, p_y^0, \theta^0\} \leftarrow$ SearchViaImprovedHybridA*;
8: $\boldsymbol{x}^0 \leftarrow$ Transform($p_x^0, p_y^0, \theta^0$);
9: Second stage: trajectory optimization
10: repeat
11:     $\Omega(\boldsymbol{x}^r) \leftarrow$ FindConvexFeasibleSets($\boldsymbol{x}^r$);
12:     $H(\boldsymbol{x}) \leftarrow$ ComputeKinematicConstraintPenalty($\boldsymbol{x}^r$);
13:     $F_{log}(\boldsymbol{x}) \leftarrow$ ComputeTerrainConstraintPenalty($\boldsymbol{x}^r$);
14:     $\boldsymbol{x}^{r+1} \leftarrow$ SolveOptimizationProblem($\boldsymbol{x}^r$);
15:     $r \leftarrow r+1$;
16:     if $r > \mathrm{r_{max}}$ then
17:         Return failure;
18:     end if
19:     $\sum F_{\max}(\boldsymbol{x}^r) \leftarrow$ MeasureInfeasibility($\boldsymbol{x}^r$);
20:     $\lambda_{r+1} \leftarrow \mathrm{c}\lambda_r$;
21: until $\sum F_{\max}(\boldsymbol{x}^r) \le \varepsilon_{\text{tol}}$ and $\left\| J(\boldsymbol{x}^{r+1}) - J(\boldsymbol{x}^r)\right\| \le \varepsilon_0$;
    return An optimal trajectory $\boldsymbol{x}^r$.
```

Requirement 1:

$$g_s(p_x(k), p_y(k)) \cdot g_c(\chi(k)) = 0, \tag{12}$$

where $g_s(p_x(k), p_y(k))$ denotes the formulation for the specific area description and $g_c(\chi(k))$ denotes the formulation for the constraint on decision variable $\chi(k)$. $\chi(k)$ refers to the decision variable constrained by the specific region, such as the vehicle orientation, steering angle, and any other variables or the combination of variables.

Requirement 2: $g_s(p_x(k), p_y(k)) \neq 0$ only when the ego vehicle is in the specific area, otherwise, $g_s(p_x(k), p_y(k)) = 0$.

To ensure (12) hold, when $g_s(p_x(k), p_y(k)) \neq 0$, $g_c(\chi(k))$ must equal zero. When $g_s(p_x(k), p_y(k)) = 0$, (12) holds and there is no constraint on $\chi(k)$.

For simplicity, we assume the specific area can be approximated by a rectangle, i.e., the area is $x_a \le p_x(k) \le x_b$ and $y_a \le p_y(k) \le y_b$. To satisfy the above requirements, the formulated $g_s(p_x(k), p_y(k))$ and $g_c(\chi(k))$ are as follows:

$$g_s\left(p_x(k), p_y(k)\right) = g_1\left(p_x(k)\right)\cdot g_2\left(p_y(k)\right), \tag{13}$$

where

$$g_1(p_x(k)) = \max(-(p_x(k)-x_a)(p_x(k)-x_b), 0), \tag{14}$$

$$g_2(p_y(k)) = \max(-(p_y(k)-y_a)(p_y(k)-y_b), 0), \tag{15}$$

and

$$g_c(\chi(k)) = \max\left(\left(\chi_a - \chi(k)\right), 0, \psi_2\left(\chi(k) - \chi_b\right)\right). \tag{16}$$

It is easy to check that our constructed expressions meet the two requirements. A visual explanation for the above

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

formulations is shown in Fig. 4. Here, (14) and (15) make the curves in Fig. 4 be quadratic curves in the upper part of the x-axis and (16) makes the curve piecewise linear.

However, the formulated spatially dependent equality constraint makes the optimization problem difficult to solve for its non-differentiability. To alleviate the non-differentiability, we adopt the smooth *logsumexp* function for approximation, since

$$\max(x_1, x_2, \ldots, x_\mathrm{m}) \approx \frac{1}{\psi}\log\left(\sum_{i=1}^{\mathrm{m}} e^{\psi x_i}\right). \tag{17}$$

Here, x_i denotes an arbitrary real number and ψ is called the smoothing parameter. $\psi > 0$ and the approximation effect improves as ψ increases [28]. We let

$$\begin{aligned} F_{\max}(\boldsymbol{x}(k)) = &\max(-(p_x(k)-x_a)(p_x(k)-x_b), 0)\\ &\cdot\max(-(p_y(k)-y_a)(p_y(k)-y_b), 0)\\ &\cdot\max((\chi_a-\chi(k)), (\chi(k)-\chi_b), 0), \end{aligned} \tag{18}$$

and its smooth function approximation can be expressed as

$$\begin{aligned} F_{\log}(\boldsymbol{x}(k)) = &\frac{1}{\psi}\log(e^{-\psi(p_x(k)-x_a)(p_x(k)-x_b)}+1)\\ &\cdot\frac{1}{\psi}\log(e^{-\psi(p_y(k)-y_a)(p_y(k)-y_b)}+1)\\ &\cdot\frac{1}{\psi}\log(e^{\psi(\chi_a-\chi(k))}+e^{\psi(\chi(k)-\chi_b)}+1). \end{aligned} \tag{19}$$

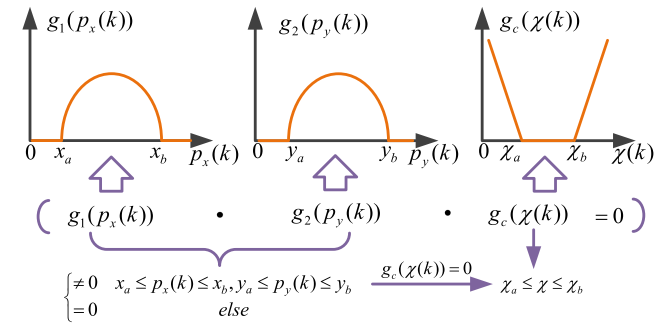
We use $F_{\max}(\boldsymbol{x})$ to denote $[F_{\max}(\boldsymbol{x}(0)), \ldots, F_{\max}(\boldsymbol{x}(n))]^\mathrm{T}$ and $F_{\log}(\boldsymbol{x})$ to denote $[F_{\log}(\boldsymbol{x}(0)), \ldots, F_{\log}(\boldsymbol{x}(n))]^\mathrm{T}$ for simplicity. $F_{\max}(\boldsymbol{x}) = 0$ requires that every element in vector $F_{\max}(\boldsymbol{x})$ equal zero and $\sum F_{\max}(\boldsymbol{x})$ denotes the sum of all elements.

To alleviate the difficulty brought by the nonlinearity of constraints, we punish (19) to the cost function as $J(\boldsymbol{x}) + \lambda_r F_{\max}(\boldsymbol{x})$. The penalty parameter $\lambda_r > 0$ is an increasing sequence of constants satisfying $\lambda_{r+1} = \mathrm{c}\lambda_r$ until $\sum F_{\max}(\boldsymbol{x}) \le \varepsilon_{\mathrm{tol}}$, where c is the growth coefficient and $\varepsilon_{\mathrm{tol}}$ is the convergence tolerance for spatially dependent constraints.

Empirically, a proper selection of the penalty factor is vital for robust and safe planning and should be tuned in the simulation until a satisfactory performance. The proposed handling method for spatially dependent constraints is beneficial to the real-time solution of the trajectory planning optimization problem and can be easily extended to arbitrary expressions of decision variables.

B. First Stage: Coarse Trajectory Searching

(19) increases the OCP non-convexity, requiring a proper initial guess to warm-start the numerical optimization solution process. In the first stage, SearchViaImprovedHybridA*() in line 7 of Alg. 1 is used to provide such guess satisfying spatially dependent constraints. The classic hybrid A* algorithm [29] uses the Reed-Shepp model [30] to expand children nodes in analytical expansions, and children nodes are only added if they are collision-free [29]. We propose an improved hybrid A* (IHA) algorithm by restricting node expansion. When the trajectory passes through a specific region, the node



$g_1(p_x(k))$
$g_2(p_y(k))$
$g_c(\chi(k))$
0 x_a x_b $p_x(k)$
0 y_a y_b $p_y(k)$
0 χ_a χ_b $\chi(k)$
$\left(g_1(p_x(k)) \cdot g_2(p_y(k)) \cdot g_c(\chi(k)) = 0 \right)$
$\begin{cases}\neq 0 & x_a \le p_x(k) \le x_b, y_a \le p_y(k) \le y_b\\ = 0 & else\end{cases}$
$g_c(\chi(k)) = 0$
$\chi_a \le \chi \le \chi_b$

Fig. 4. An illustration of our formulation for spatially dependent constraints.

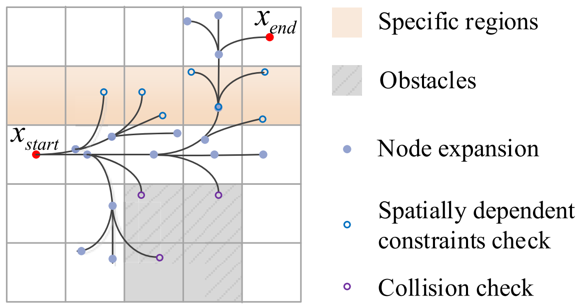


Fig. 5. The node expansion diagram of our improved hybrid A* algorithm.

expansion process checks for spatially dependent constraint satisfaction. Only when the node is collision-free and it does not violate the spatially dependent constraints, the node is added to the graph. Our improved node expansion is shown in Fig. 5.

It is worth noting, in line 6 of Alg. 1, DilateObstacle() dilates the obstacles on the grid map to include the size of the ego vehicle, so that the vehicle can be considered as a mass point, simplifying collision detection [20].

The trajectory found by the IHA only involves decision variables $\{p_x^0, p_y^0, \theta^0\}$. Transform() gets the values of v^0 and φ^0 by the following equations, where $v^0(n)$ and $\varphi^0(n)$ are user-defined values:

$$v^0(k) = \frac{\left\| P^0(k+1) - P^0(k) \right\|_2}{t_n^0/n}, \quad k = 0, \ldots, n-1, \tag{20}$$

$$\varphi^0(k) = \tan^{-1}\left(\frac{\theta^0(k+1)-\theta^0(k)}{t_n^0/n}\frac{\mathrm{L_m}}{v^0(k)}\right), \quad k = 0, \ldots, n-1. \tag{21}$$

Here, t_n^0 is set based on the size of the configuration space and $v_{\max}$. In this way, the initial trajectory $\boldsymbol{x}^0 = \{p_x^0, p_y^0, \theta^0, v^0, \varphi^0, t_n^0\}$ is obtained. We use the initial guess $\boldsymbol{x}^0$ to warm-start the second stage numerical OCP solution process.

C. Second Stage: Trajectory Optimization

The trajectory optimization problem is non-convex due to constraints introduced by obstacles and vehicle kinematics.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PARAMETRIC SETTINGS

Symbol	Description	Parameter
L	vehicle length	4.5 m
W	vehicle width	2 m
L_m	vehicle wheelbase	2.5 m
$v_{\min}$	minimum velocity	0 m/s
$v_{\max}$	maximum velocity	2.5 m/s
$a_{\min}$	minimum acceleration	-2 m/s^2
$a_{\max}$	maximum acceleration	2 m/s^2
$\varphi_{\min}$	minimum steering angle	-0.7 rad
$\varphi_{\max}$	maximum steering angle	0.7 rad
ε_0	termination threshold in our algorithm	10^{-3}
ε_{tol}	convergence tolerance for violation of nonlinear equality constraints	10^{-3}
$r_{\max}$	maximum number of iterations in our algorithm	200
λ_0	initial value for the terrain constraint penalty parameter	10^4
c	growth coefficient for the penalty parameter	4

To facilitate the numerical optimization solution process, we convert the non-convex problem to a series of convex problems by adopting the following two commonly used tricks. For the non-convex collision-avoidance constraints, FindConvexFeasibleSets() in line 11 of Alg. 1 adopts the convex feasible set algorithm [31] to find the convex corridor $\boldsymbol{\Omega}(\boldsymbol{x}^r)$ around each waypoint based on the solution of the r^{th} iteration. For the nonlinear kinematic equations, ComputeKinematicConstraintPenalty() linearizes them sequentially and penalizes the linearized constraints to the cost function with the L1 norm as $H(\boldsymbol{x})$ [23]. Due to the limitation of space, we do not introduce this part of the treatment in detail, interested readers please refer to [31] and [23]. In line 13 of Alg. 1, ComputeTerrainConstraintPenalty() computes $F_{log}(\boldsymbol{x})$ for spatially dependent constraint penalty. Thus, the optimization problem in the r^{th} iteration can be formulated as follows:

$$\begin{aligned}&\min_{x} \;\; J(\boldsymbol{x}) + H(\boldsymbol{x}) + \lambda_r F_{log}(\boldsymbol{x})\\ &s.t. \; (4)-(7), \;\; \boldsymbol{x} \in \boldsymbol{\Omega}(\boldsymbol{x}^r).\end{aligned} \tag{22}$$

Considering the merits of high computational efficiency and high solution quality of the primal-dual interior-point method [32], [33], SolveOptimizationProblem() adopts the primal-dual interior-point method to solve (22). It uses the result of the previous iteration to warm-start the next solution process. Our algorithm can guarantee convergence, whose proof process is similar to [31] and [23]. We do not present these proofs in this paper due to the limitation of space.

IV. NUMERICAL RESULTS

This section illustrates the effectiveness of our proposed trajectory planning framework. The simulations are carried out in Matlab R2018a (Win64) on a laptop, with the processor Intel(R) Core(TM) i5-8400 CPU@2.80GHz. Basic parametric settings are summarized in TABLE I. The xoy and heading resolution for the IHA is 0.5 m and $\pi/6$, respectively.

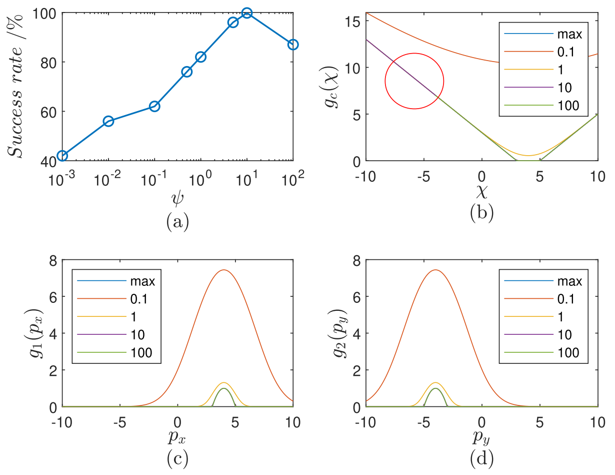


Fig. 6. Success rate and function approximation under different settings of ψ: (a) success rate, (b) approximation for $g_c(\chi(k))$, (c) approximation for $g_1(p_x(k))$, and (d) approximation for $g_2(p_y(k))$. It should be mentioned, (b)-(d) are just one representative sample of the 100 random cases.

A. Parameter Analysis

In our proposed algorithm, ψ is a critical parameter that decides the satisfaction of spatially dependent constraints. To determine the proper set of the smoothing parameter ψ, we manually generate 100 random cases, where the width and the length of the specific area are both changed from 2 m to 8 m within a 40 m × 40 m workspace. The initial and terminal locations of the ego vehicle are uniformly distributed in the workspace while ensuring that it must pass through a local specific area. The 100 cases are classified according to spatially dependent constraints on vehicle speed, steering, and orientation.

By setting ψ to different values, the success rate for finding a feasible trajectory with spatially dependent constraints satisfied is depicted in Fig. 6 (a). Fig. 6 (a) shows that the success rate first increases and then decreases. As seen from Fig. 6 (b)-(d), a small ψ cannot provide a precision approximation to the original constraint. However, a large ψ increases the non-differentiability and may make the calculated value too large and exceed the computer's numerical processing upper bound, e.g., in Matlab, $e^{1000} = Inf, log(e^{1000}) = Inf$. As shown in Fig. 6 (b), the values in the red circle for $\psi = 100$ are infinitely large, which may make the solution fail to jump from the bad local optimum. Thus, in the rest of this paper, we set $\psi = 10$.

B. Comparisons With Existing Planners

This subsection designs a simple case to illustrate the advantages of the proposed method over other planners. In this case study, an autonomous vehicle traverses a slope with $h = 1$ m and $\alpha = 6°$ as shown in Fig. 2. Herein, we are not concerned with no-tip-over and friction constraints, which are mainly for all-steering off-road vehicles in mountain terrains. Instead, we only focus on the vehicle orientation limits so that falling from the slope can be prevented. By setting an interval $[\pi/3, 2\pi/3]$ for $\theta(k)$, we want to ensure that the vehicle does not fall down the slope. The initial and terminal

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

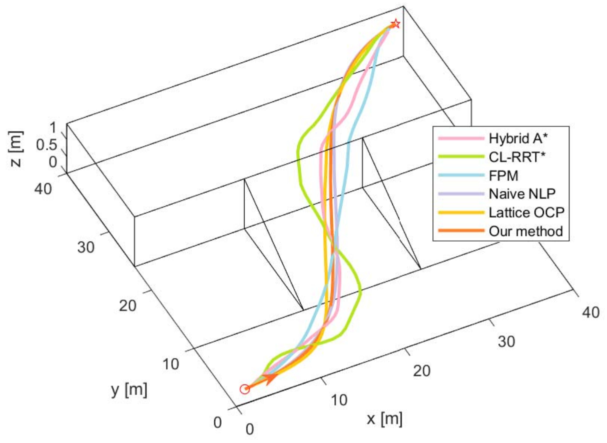


Fig. 7. Planned trajectories of comparative planners in case 1.

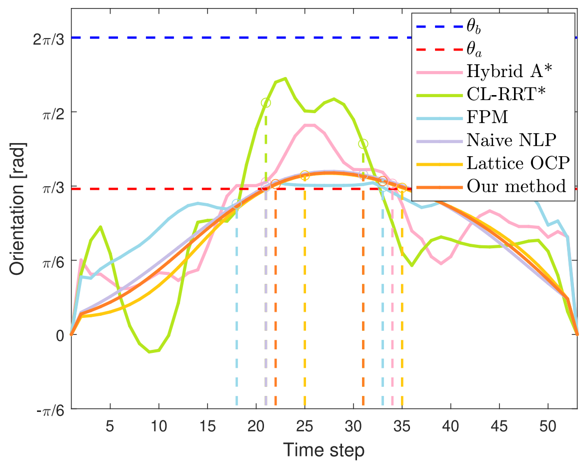


Fig. 8. Variations of the orientation angle with the time step found by the comparative methods in Case 1. Herein, circles represent the starting and ending time instances for the ego vehicle to traverse the slope, and θ_a and θ_b denote the minimum and maximum orientation limits in the slope area.

positions of the ego vehicle in the 3D coordinate system are assigned to $[2, 2, 0]^{\mathrm{T}}$ and $[38, 38, 1]^{\mathrm{T}}$, respectively. The spatially dependent constraint for this case is described as follows.

Case 1 (Trajectory Planning for Traversing a Slope): Spatially Dependent Constraint: if the ego vehicle passes through the slope, i.e., $13 \leq p_x(k) \leq 27$ and $10 \leq p_y(k) \leq 10 + h/\sin\alpha$, then it is required that $\pi/3$ and $2\pi/3$.

The planned trajectories of the proposed method and other comparable planners are shown in Fig. 7. The corresponding variations of the orientation angle with the time step are shown in Fig. 8. The number of total time steps for all planners is set to 53. The comparable planners are defined in TABLE II and their comparative performances are summarized in TABLE III. Here, the trajectory feasibility judgment in TABLE III is based on whether the waypoints on the slope satisfy the orientation angle limits, which can be seen from Fig. 8.

We adopt two representative sampling-and-search-based planners, i.e., the hybrid A* algorithm [29] and the closed-loop RRT* (CL-RRT*) algorithm [34], and three

TABLE II
DEFINITION OF COMPARATIVE PLANNERS

Method	Description
Hybrid A*	Improve the original hybrid A* algorithm [R1] by adding nodes satisfying spatially dependent terrain constraints.
CL-RRT*	Improve the original CL-RRT* algorithm [34] by adding samples satisfying spatially dependent terrain constraints.
FPM	Only impose terrain constraints on fixed points in the specific area [6] based on the path derived from the hybrid A* algorithm.
Naïve NLP	Solve the nominal OCP numerically [35], where the initial guess and the spatially dependent constraint handling are the same as ours.
Lattice-OCP	Solve the nominal OCP numerically, where the initial guess is provided by lattice [19] and the spatially dependent constraint handling are the same as ours.

TABLE III
PERFORMANCE COMPARISONS

Method	Solution time	Trajectory cost	Feasibility
Hybrid A*	0.14 s	10783.92	Yes
CL-RRT*	0.08 s	24215.76	Yes
FPM	0.26 s	8345.12	No
Naïve NLP	3.68 s	2710.34	Yes
Lattice OCP	2.88 s	2714.56	Yes
Ours	0.32 s	2712.42	Yes

optimization-based planners for comparison. When using the sampling-and-search-based planners, the node expansion process checks for spatially dependent constraint satisfaction. For the optimization-based planners, Naïve NLP [35] and Lattice OCP [19], we take the terrain constraint handling method stated in this paper for a fair comparison.

Featured by only searching, the hybrid A* algorithm performs poorly in terms of the trajectory cost since it contains unnatural swerves due to curvature discontinuity. Its generated trajectory is not smooth and deserves further improvement with optimization.

The closed-loop RRT* algorithm is well-known for quickly deriving a feasible trajectory, but the generated trajectory swerves, as reflected by the green line in Fig. 7, which is not optimal. Increasing the number of iterations may enhance the optimality, but that leads to a heavier computational burden.

The fixed-point method (FPM) [6] only imposes spatially dependent constraints on the waypoints on the slope based on the solution from the hybrid A* algorithm. Although it takes the least time among optimization-based methods, it cannot ensure the trajectory feasibility since the waypoints on the slope will change after optimization. As shown in Fig. 8, the waypoints on the slope found by the hybrid A* algorithm are from step 21 to step 34 and FPM only imposes orientation constraints on these steps. However, the final steps in the specific slope area found by FPM are from step 17 to step 33, making the waypoints from step 17 to step 21 violate the slope constraint on the vehicle orientation.

Naïve NLP and Lattice OCP impose spatially dependent constraints on all waypoints like ours. They can obtain smooth and safe trajectories while satisfying spatially dependent constraints. However, it is time-consuming to solve the nominal

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

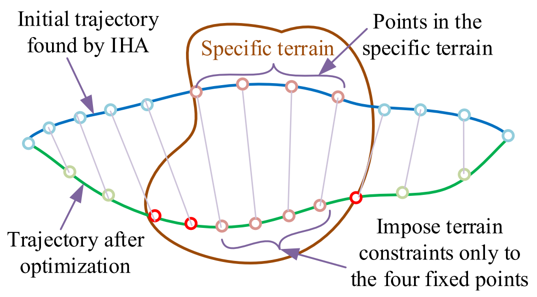

Initial trajectory found by IHA
Specific terrain
Points in the specific terrain
Trajectory after optimization
Impose terrain constraints only to the four fixed points

Fig. 9. A schematic of the fixed-point method, which only imposes terrain constraints to fixed points based on the solution from the IHA algorithm. However, after the optimization solution, the points in the specific area change. The fixed-point method cannot ensure the safety of the newly added points, like the red dots in the figure.

OCP directly (Naïve NLP) without simplifying the non-convex collision avoidance constraints and nonlinear kinematic constraints. Lattice OCP adopts a high-quality initial guess from a lattice-based planner which considers vehicle kinematics for the subsequent numerical optimization process. Since it spends much time seeking the initial guess among numerous motion primitives, its total solution time is longer than ours.

By contrast, the proposed method overcomes the aforementioned drawbacks. First, it optimizes all waypoints by numerically solving an OCP, which greatly improves the trajectory smoothness and quality. Second, it imposes spatially dependent constraints handling on all waypoints, which can ensure trajectory safety when the waypoints in the specific area change. Third, the proposed method converts the original OCP to a series of optimization problems by using convex feasible sets and softening the kinematic constraints to the cost function, which greatly facilitates the numerical OCP solution process.

In summary, compared with other state-of-art planners, ours can achieve accurate, fast, and high-quality performance for 2.5D trajectory planning.

C. Analysis of Spatially Dependent Constraint Handling

To illustrate the effect of our spatially dependent constraint handling method, we compare it with other strategies. The simulation settings are as in IV. B. Here are the strategy descriptions. The *Max equality* strategy and the *Logsumexp equality* strategy solve the optimization problems with the equality constraint $F_{max}(\boldsymbol{x}) = 0$ and $F_{log}(\boldsymbol{x}) = 0$, respectively. In contrast, the proposed method penalizes $F_{log}(\boldsymbol{x})$ onto the cost function. For comparison, the three optimization problems are all solved by the *fmincon* function built-in Matlab, which adopts subgradient methods to deal with non-differentiability [36].

TABLE IV shows the results of the three strategies for handling spatially dependent constraints. If the obtained solution satisfies $\sum F_{\max}(\boldsymbol{x}) \le \varepsilon_{\text{tol}}$, it is considered as a feasible trajectory satisfying spatially dependent constraints, where $\varepsilon_{\text{tol}} = 10^{-3}$. As seen in TABLE IV, the trajectories obtained from the three strategies are all feasible and their trajectory costs are similar.

TABLE IV
STRATEGY PERFORMANCE COMPARISONS

Strategy	Description	Runtime	Cost	Feasibility
Max equality	$F_{max}(\boldsymbol{x}) = 0$	0.62 s	1928.12	Yes
Logsumexp equality	$F_{log}(\boldsymbol{x}) = 0$	0.67 s	1965.25	Yes
Logsumexp penalty	$J(\boldsymbol{x}) + \lambda F_{log}(\boldsymbol{x})$	0.34 s	1923.57	Yes

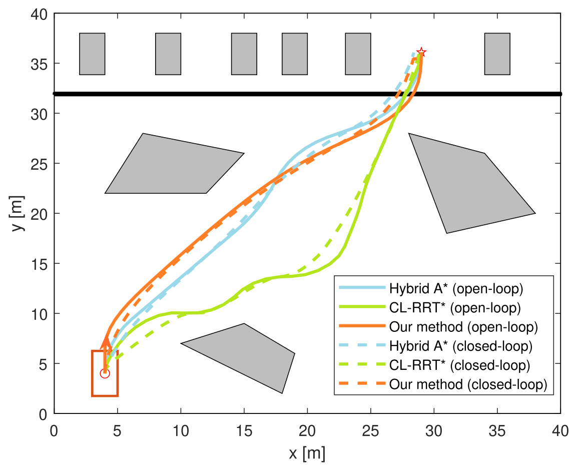


Fig. 10. Comparative open-loop trajectory planning and closed-loop trajectory tracking simulation results in Case 2. Herein, the black line represents the edge of the road shoulder.

Compared with handling spatially dependent constraints as equality constraints, ours can greatly reduce runtime since it simplifies the to-be-solved optimization problem. The results in TABLE IV validate the high solution speed of our spatially dependent handling method.

D. Vehicle Dynamics Simulation Validation

This subsection explores the dynamic feasibility of our proposed trajectory planner by a challenging shoulder case. In this case, an autonomous vehicle needs to drive from the road to the shoulder with a height of 15 cm and avoid collision with surrounding obstacles, as shown in Fig. 10. The initial and terminal positions of the ego vehicle in the 3D coordinate system are assigned to $[4, 4, 0]^{\text{T}}$ and $[29, 36.1, 0.15]^{\text{T}}$, respectively. When the vehicle is approaching the shoulder curb, it is necessary to try to keep the tires perpendicular to the shoulder. To achieve this goal, we restrict the vehicle orientation $\theta(k)$ and the angle between the steering angle and the shoulder $\tau(k)$ to between $\pi/4$ and $3\pi/4$ when the ego vehicle is approaching the shoulder curb. The spatially dependent constraint for this case is described as follows.

Case 2 (Trajectory Planning for Traversing a Shoulder): Spatially Dependent Constraint: if the ego vehicle passes through the shoulder, i.e., $31.85 \le p_y(k) \le 32$, then it is required that $\pi/4 \le \theta(k) \le 3\pi/4$ and $\pi/4 \le \tau(k) \le 3\pi/4$.

To further illustrate the dynamic feasibility of our planned trajectory, we compare the proposed method mainly with sampling-and-search-based planners, considering optimization-based planners obtain relatively similar trajectories, making the differences in the tracking results not clear.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

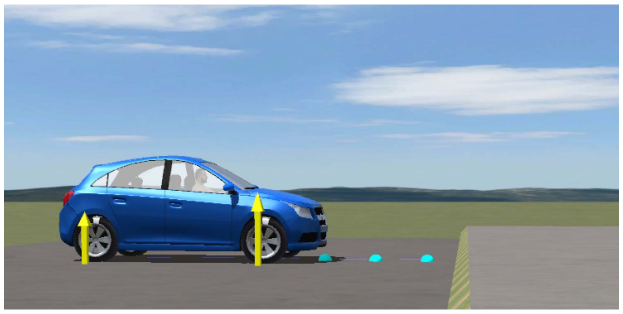


Fig. 11. Visualization of the ego vehicle during its closed-loop tracking process in the CarSim's simulator.

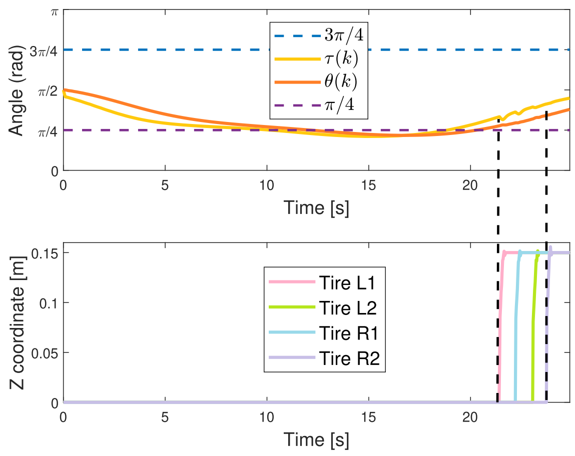


Fig. 12. Variations of $\tau(k)$, $\theta(k)$, and ground z coordinates of the four tires in case 2.

This can be corroborated by the trajectories derived from Naïve NLP, Lattice OCP, and ours in Fig. 7. The dynamics test is performed in CarSim, which is shown in Fig. 11. The road shoulder is approximated by two connected planes with a height difference of 15 cm, whose friction coefficient is set to 0.85. The simulated vehicle model is *C-Class, Hatchback*, whose parameters are set by default except for those mentioned in Table I. In CarSim, we adopt the *steer from driver model* [37] in the closed-loop control module to track the trajectory. The driver preview time is set to 5 s and the execution frequency of the control module is set to 40.0 Hz. The planning and tracking results of comparative planners are shown in Fig. 10.

Fig. 10 shows that our planned trajectory is more smooth and more natural w.r.t the vehicle kinematics than the trajectories derived from the hybrid A* algorithm and the CL-RRT* algorithm, minimizing closed-loop trajectory tracking errors. Since the adopted motion primitives in the hybrid A* algorithm can only guarantee the generated path to be orientation continuous, the curvature profile of the path may discontinue at the connection points of different motion primitives. The problem of curvature discontinuity increases the closed-loop trajectory tracking error in the hybrid A* algorithm. The trajectory generated by the CL-RRT* algorithm has many swerves, making it hard to be tracked. Increasing the number of iterations can reduce the trajectory swerves and increase the trajectory smoothness, but that would impose a greater computational burden.

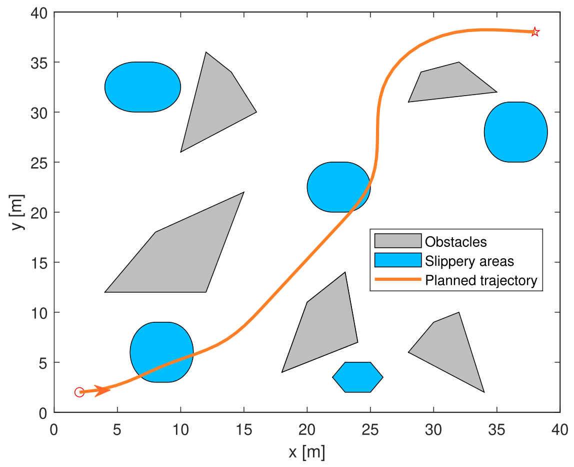


Fig. 13. The planned trajectory for the ego vehicle traversing many slippery areas in case 3.

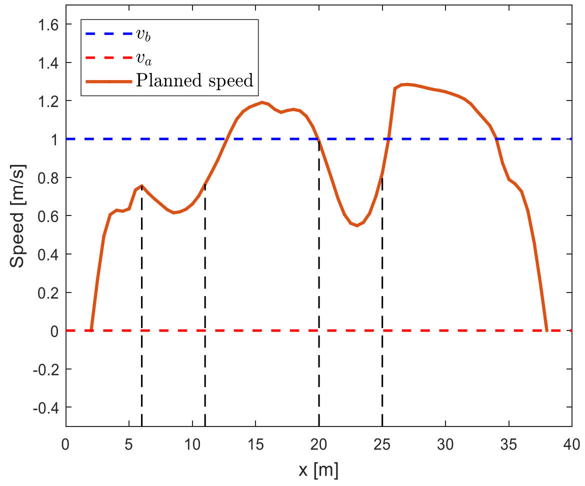


Fig. 14. The variation of the planned speed with the horizontal coordinate of the ego vehicle in case 3. Herein, v_a and v_b denote the minimum and maximum speed limits in the slippery areas.

To observe the variations of $\theta(k)$ and $\tau(k)$ during the trajectory tracking process, we draw it with the variation of the ground z coordinates of the four tires. Fig. 12 shows that the z coordinates of the ego vehicle's four wheels change as the vehicle approaches the shoulder between 21.42 s and 23.75 s. During this time, the angle $\tau(k)$ and the vehicle orientation $\theta(k)$ are always between $\pi/4$ and $3\pi/4$, that is, our planned trajectory can make the vehicle meet the spatially dependent constraints.

E. Algorithm Validation on Compound Restricted Areas

This subsection considers a more complex case to demonstrate the applicability of our proposed framework. In contrast to the previous examples, it has many restricted areas. As shown in Fig. 13, there are many slippery areas. It is required that the vehicle speed be less than 1 m/s once it enters a slippery area to prevent the wheels from skidding

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

due to excessive speed. The initial state $\boldsymbol{x}_{\text{start}}$ and the terminal state $\boldsymbol{x}_{\text{end}}$ of the ego vehicle are set as $[2, 2, 0, 0, 0]^{\mathrm{T}}$ and $[38, 38, 0, 0, 0]^{\mathrm{T}}$, respectively. For simplicity, we assume that each slippery area can be covered by a rectangle of a minimum area and each rectangle can be represented as $x_a^s \leq x \leq x_b^s$ and $y_a^s \leq y \leq y_b^s$. Here, s represents the s^{th} slippery area and there are ns slippery areas in total. The spatially dependent constraint for this case is described as follows.

Case 3 (Trajectory Planning for Traversing Multiple Restricted Areas): Spatially Dependent Constraint: if the ego vehicle traverses a slippery area, i.e., $x_a^s \leq p_x(k) \leq x_b^s$ and $y_a^s \leq p_y(k) \leq y_b^s$, for $s = 1, \ldots, n\ s$, then it is required that $0 \leq v(k) \leq 1$ m/s.

The planned trajectory and the corresponding speed profile are shown in Fig. 13 and Fig. 14, respectively. On its way towards the goal pose, the vehicle traverses two slippery areas with horizontal coordinates ranging from 6 m to 11 m and 20 m to 25 m, respectively. As a result of the initial and terminal speeds being set to zero, the speed profile exhibits an upward trend when the vehicle approaches the first slippery area and a downward trend when it leaves the second slippery area. Once the ego vehicle enters the slippery areas, the spatially dependent constraint w.r.t. v_k takes effect, making v_k constrained in the interval [0, 1 m/s]. This result illustrates the applicability of the proposed framework in a challenging scenario with multiple restricted areas.

V. Conclusion

In this paper, a two-stage trajectory planning framework is proposed for 2.5D planning problems with spatially dependent constraints. To effectively handle the spatially dependent constraints with an "if-else" structure, an approximation formulation for these constraints is designed to eliminate the non-differentiability of these constraints. In this way, the second stage gradient-based optimizer can quickly solve the trajectory optimization problem with an initial guess from the first stage IHA, thereby improving the computational efficiency as well as the trajectory quality. This conclusion is verified by several typical case studies and comparative simulations.

It deserves to point out that the two-stage optimization solution is highly dependent on the quality of the initial guess provided from the first stage. If the adopted hybrid A* algorithm gives a misleading initial coarse trajectory to the optimizer, the proposed algorithm may be stuck in a poor local optimum. In the future, we will design a fault-recovery strategy to enhance the stability of the proposed framework and improve the solution speed for better practical performance.

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. 2019.

[2] M. Li *et al.*, "A two-layer potential-field-driven model predictive shared control towards driver-automation cooperation," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 29, 2021, doi: 10.1109/TITS.2020.3044666.

[3] C. Wang, M. Xia, and M. Q.-H. Meng, "Stable autonomous robotic wheelchair navigation in the environment with slope way," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 10759–10771, Oct. 2020.

[4] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3D path planning and execution for search and rescue ground robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 722–727.

[5] M. Cherif, "Motion planning for all-terrain vehicles: A physical modeling approach for coping with dynamic and contact interaction constraints," *IEEE Trans. Robot. Autom.*, vol. 15, no. 2, pp. 202–218, Apr. 1999.

[6] R. Usami, Y. Kobashi, T. Onuma, and T. Maekawa, "Two-lane path planning of autonomous vehicles in 2.5 D environments," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 2, pp. 281–293, Nov. 2019.

[7] D. F. Rudny and D. W. Sallmann, "Analysis of accidents involving alleged road surface defects (IE, shoulder drop-offs, loose gravel, bumps and potholes)," SAE, Warrendale, PA, USA, Tech. Rep., 1996.

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

[9] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[10] J. Gu and Q. Cao, "Path planning for mobile robot in a 2.5-dimensional grid-based map," *Ind. Robot: Int. J.*, vol. 38, no. 3, pp. 315–321, May 2011.

[11] D. Devaurs, T. Siméon, and J. Cortés, "Optimal path planning in complex cost spaces with sampling-based algorithms," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 415–424, Apr. 2016.

[12] N. Ganganath, C.-T. Cheng, and C. K. Tse, "A constraint-aware heuristic path planner for finding energy-efficient paths on uneven terrains," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 601–611, Jun. 2015.

[13] 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.

[14] J. Chen, W. Zhan, and M. Tomizuka, "Autonomous driving motion planning with constrained iterative LQR," *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 244–254, Jun. 2019.

[15] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—A local, continuous method," in *Proc. IEEE Intell. Vehicles Symp. Proc.*, Jun. 2014, pp. 450–457.

[16] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Multiobjective optimal parking maneuver planning of autonomous wheeled vehicles," *IEEE Trans. Ind. Electron.*, vol. 67, no. 12, pp. 10809–10821, Dec. 2020.

[17] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "Achieving real-time path planning in unknown environments through deep neural networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2093–2102, Mar. 2022.

[18] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. L. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 17, 2020, doi: 10.1109/TNNLS.2020.3042120.

[19] K. Bergman, O. Ljungqvist, and D. Axehill, "Improved path planning by tightly combining lattice-based path planning and optimal control," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 1, pp. 57–66, Mar. 2021.

[20] B. Li *et al.*, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Trans. Intell. Transp. Syst.*, early access, Sep. 8, 2021, doi: 10.1109/TITS.2021.3109011.

[21] F. Tian *et al.*, "Trajectory planning for autonomous mining trucks considering terrain constraints," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 4, pp. 772–786, Dec. 2021.

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

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

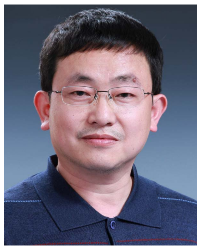
[24] H. Cao *et al.*, "An optimal hierarchical framework of the trajectory following by convex optimisation for highly automated driving vehicles," *Vehicle Syst. Dyn.*, vol. 57, no. 9, pp. 1287–1317, Sep. 2019.

[25] Y. Guo, H. Xu, D. Yao, and L. Li, "A real-time optimization-based trajectory planning method in dynamic and uncertain environments," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.

[26] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.
[27] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Trans. Robot.*, vol. 36, no. 5, pp. 1526–1545, Oct. 2020.
[28] X. Tong, F. F. Wu, and L. Qi, "Available transfer capability calculation using a smoothing pointwise maximum function," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 1, pp. 462–474, Feb. 2008.
[29] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.
[30] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, Oct. 1990.
[31] C. Liu, C. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM J. Control Optim.*, vol. 56, no. 4, pp. 2712–2733, Jun. 2018.
[32] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia, PA, USA: SIAM, 1997.
[33] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM J. Optim.*, vol. 2, no. 4, pp. 575–601, 1992.
[34] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.
[35] 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.
[36] R. H. Byrd, M E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM J. Optim.*, vol. 9, no. 4, pp. 877–900, 1999.
[37] C. C. MacAdam, "Application of an optimal preview control for simulation of closed-loop automobile driving," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. SMC-11, no. 6, pp. 393–399, Jun. 1981.

Yuqing Guo received the B.S. degree from the Harbin Institute of Technology, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Automation, Tsinghua University, China. Her current research interests include autonomous driving, cooperative driving, and trajectory planning.



Danya Yao (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Tsinghua University, Beijing, China, in 1988, 1990, and 1994, respectively. He is currently a Full Professor with the Department of Automation, Tsinghua University. His research interests include intelligent detection technology, system engineering, mixed traffic flow theory, and intelligent transportation systems.

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, USA, as a Joint Training Ph.D. Student. He is currently an Associate Professor with the College of Mechanical and Vehicle Engineering, Hunan University, China. Prior to teaching at Hunan University, he worked at the JDX Research and Development Center of Automated Driving, JD Inc., China, from 2018 to 2020, as an Algorithm Engineer. He has been the first author of 60 journal/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.

Zimin He is currently pursuing the bachelor's degree with the Department of Automation, Tsinghua University, Beijing, China. His research focuses on convex optimization and trajectory planning.

Haichuan Gao received the B.S. degree from the Department of Mechanics, Central South University, China, in 2018. He is currently pursuing the Ph.D. degree in control science and engineering with Tsinghua University. His research focuses on robotic artificial intelligence and reinforcement learning.

Li Li (Fellow, IEEE) is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China, working in the fields of artificial intelligence, complex systems, intelligent control and sensing, intelligent transportation systems, and intelligent vehicles. He has published over 100 SCI indexed international journal articles and over 70 international conference papers as a first/corresponding author. He is a member of the Editorial Advisory Board for the *Transportation Research Part C: Emerging Technologies* and a member of the Editorial Board for the *Transport Reviews* and *ACTA Automatica Sinica*. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

Authorized licensed use limited to: HEFEI UNIVERSITY OF TECHNOLOGY. Downloaded on October 29,2024 at 13:36:08 UTC from IEEE Xplore. Restrictions apply.