An Efficient Spatial-Temporal Trajectory Planner for Autonomous Vehicles in Unstructured Environments

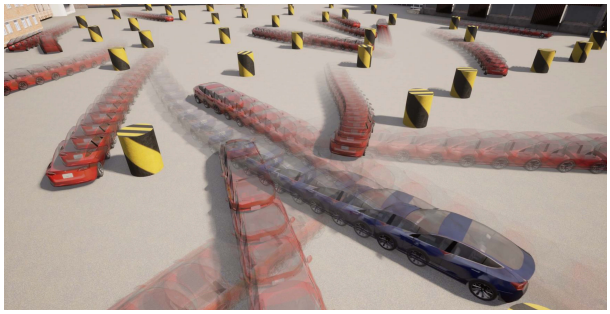
Zhichao Han*[1,2], Yuwei Wu*[3], Tong Li[4], Lu Zhang[4], Liuao Pei[1,2], Long Xu[1,2], Chengyang Li[2,5], Changjia Ma[1,2], Chao Xu[1,2], Shaojie Shen[4], and Fei Gao[1,2]

***Abstract*—As a fundamental component of autonomous driving systems, motion planning has garnered significant attention from both academia and industry. This paper focuses on efficient and spatial-temporal optimal trajectory optimization in unstructured environments using compact convex approximations of vehicle shapes. Conventional approaches typically model the task as an optimal control problem by discretizing the motion process in state configuration space. However, this often results in a tradeoff between optimality and efficiency since generating high-quality motion trajectories often requires high-precision discretization of the dynamic process, which imposes a substantial computational burden. To address this issue, we leverage the differential flatness property of car-like robots to simplify the trajectory representation and analytically formulate the spatial-temporal joint optimization problem with flat outputs in a compact manner, while ensuring the feasibility of nonholonomic dynamics. Moreover, we achieve efficient obstacle avoidance with a collision-free driving corridor for unmodelled obstacles and signed distance approximations for dynamic moving objects. We present comprehensive benchmarks with State-of-the-Art methods, demonstrating the significance of the proposed method in terms of efficiency and trajectory quality. Real-world experiments verify the practicality of our algorithm. We will release our codes for the research community.**

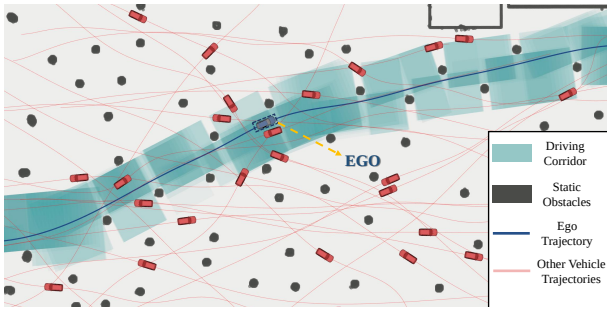
***Index Terms*—Autonomous Vehicles: Motion Planning, Trajectory Optimization, Collision Avoidance.**

I. Introduction

AUTONOMOUS driving has become one of the hottest research topics in recent years because of its vast potential social benefits. It reveals a huge demand for robust and collision-free motion planning in complex and high-dynamic environments. Lightweight and efficiency are strongly demanded in real-world applications to ensure rapid response to dynamic and unstructured environments with limited onboard computing power. Motion planning for autonomous driving aims to generate a comfortable, low-energy, and physically feasible trajectory that makes the ego vehicle reach end states with collision-free guarantees and designed velocities in constrained environments. Classical trajectory planning methods can be broadly categorized into two main groups: sampling-based and optimization-based. Sampling-based methods typically generate samples in a discretized state space and evaluate the cost function of these samples to obtain the path. Conversely, optimization-based methods formulate trajectory planning as an optimization problem and utilize gradient-based numerical solvers to find the optimal solution. While sampling-based methods are theoretically resolution-complete and can explore the entire state space to obtain the optimal solution, this often requires computationally expensive resources. Therefore, in this paper, we focus on optimization-based methods, which can achieve accurate and effective convergence to local optimal solutions by allowing trajectories to have strong deformation abilities in continuous space. How-



(a) The movement diagram.



(b) The trajectory visualization.

Fig. 1: The two figures show the capability of our planner in a highly dynamic environment. Because of full-dimensional obstacle avoidance, the ego vehicle has the ability to pass through tight areas near other moving objects while ensuring collision-free property.

This work was supported by the DJI-ZJU FAST Autonomous Drone Research Funding, and the Fundamental Research Funds for the Central Universities. (*Corresponding author: Fei Gao*)

* co-first authors.

[1]State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China.

[2]Huzhou Institute of Zhejiang University, Huzhou 313000, China.

[3]Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA.

[4]Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong.

[5]Robotics and Autonomous Systems Thrust, Systems Hub, The Hong Kong University of Science and Technology (Guangzhou), Nansha, Guangzhou, 511453, Guangdong, China.

Yuwei Wu contributes to this work during internship at Zhejiang University.

E-mail: { zhichaohan, fgaoaa}@zju.edu.cn

ever, generating feasible and high-quality trajectories online in arbitrarily complex scenarios is still challenging. In fact, ideal motion planning for autonomous driving typically faces three challenging problems:

1) **Nonholonomic Dynamics**: Unlike holonomic robots such as omnidirectional mobile robots and quadrotors, autonomous vehicles must consider nonholonomic constraints during trajectory planning. Moreover, the strong non-convexity and nonlinearity of nonholonomic dynamics make it difficult to ensure the physical feasibility of states and control inputs in highly constrained environments.
2) **Precise Obstacle Avoidance**: Collision-free constraints have to balance the accuracy of object shape modeling while maintaining an affordable computation time for online vehicle (re)planning. In real-world applications, rough approximation of the ego vehicle shape, such as one or multiple circular covering of the ego vehicle, always reduces the solution space, which introduces conservativeness or even fails to find a collision-free solution in extremely cluttered areas. On the other hand, taking into account the true physical shape of obstacles in Euclidean space, which we refer to as "full-dimensional obstacle avoidance," often increases the complexity of the planning problem, resulting in a significant computational burden.
3) **Trajectory Quality**: There is always a tradeoff between computation efficiency and trajectory quality. Common methods that optimize time and space separately reduce a large partition of solution space, especially in tightly coupled spatial-temporal scenarios such as highly dynamic environments. By contrast, spatial-temporal joint optimization can fully utilize the solution space to achieve better trajectory optimality but tends to complicate the optimization problem and reduce the real-time performance.

Existing methods often fail to achieve a good balance between efficiency, high quality, and general applicability due to these intricate challenges. Some methods oversimplify the motion model [1, 2], resulting in infeasible trajectories when large turns are required. Some approaches oversimplify obstacle avoidance [3] or rely heavily on pre-defined rules [4, 5], limiting their extension to more complex and diverse scenarios. Furthermore, most methods discretize the motion process [6]–[10], making it arduous to generate top-notch trajectories within extremely limited time. In order to ensure good executability of trajectories and achieve high planning success rates in dense environments, these methods often necessitate high-precision discretization, which places significant computational demands and hampers real-time performance. Overall, there is no universal solution for trajectory planning, which makes it the Achilles' heel hindering the development of autonomous driving.

To bridge this gap, our paper aims to propose an efficient and versatile spatial-temporal trajectory planning scheme, based on compact convex approximations of vehicle shapes, that generates high-quality trajectories adhering to nonholonomic dynamics in constrained environments, as shown in Fig.1. Based on flat space, we employ piece-wise polynomials to formally parameterize trajectories and subsequently reconstruct the trajectory representation to eliminate equality constraints and simplify the optimization problem. Moreover, in our optimization problem, all feasibility constraints are encoded as analytical forms of the final reformulated optimization variables. To facilitate gradient-based numerical solutions, we analytically derive the complete gradient backpropagation chain, encompassing constraints, high-order states, flat outputs, polynomial representation space, and ultimately the reconstructed optimization variables. It is important to note that unlike omnidirectional robots, the vehicle trajectory exhibits piecewise smooth characteristics in the presence of both forward and backward motions. Here, we refer to the transition point between forward and backward motions as Gear Shifting Point (GSP). To address this issue, we use piece-wise polynomials to parameterize forward and backward movements separately. Specifically, the starting point of the next piecewise polynomial is the endpoint of the previous one, which is GSP. In contrast to previous works where the initial and final points of the piece-wise polynomials are fixed, we analyze the relationship between the GSP and the polynomial parameters. Then, we introduce Gear Shifting Point Optimization (GSPO) without introducing additional constraint conditions, which significantly improves trajectory quality, as demonstrated in ablation experiments. Besides, we use the collision-free geometric space for modeling static obstacle avoidance constraints and ensure dynamic collision-free property based on the signed distance between the ego vehicle and other dynamic objects. Benchmarks and real-world experiments verify the advanced and practical nature of our method. The main contributions of this paper can be summarized as follows:

- We present a unified and efficient trajectory planning formulation tailored for car-like robots, incorporating four essential attributes: spatial-temporal joint optimization, convex-approximation-based static and dynamic obstacle avoidance, analytical constraint expression in flat space, and efficient trajectory representation. Specifically, we extend the refined trajectory characterization to vehicles, which ensures motion continuity and reduces optimization problem dimensionality. Moreover, all constraints can be expressed by the optimization variables reformulated by the trajectory representation parameters. Furthermore, we derive the complete gradient backpropagation chain to facilitate subsequent numerical solutions.
- We analyze the characteristics of the constraints in the trajectory planning problem and reformulate the original optimization as an unconstrained one that can be effectively solved. Besides, we introduce GSPO to handle the discontinuity resulting from the switching between forward and backward motions. Our method incorporates the GSP as free optimization variables in the trajectory parameter space, thereby increasing the freedom of the trajectory and reducing dependence on the initial value, resulting in improved trajectory quality.
- In terms of engineering, we conduct extensive com-

parative experiments in simulation to demonstrate the superiority of our approach. Furthermore, we deploy our algorithm on a commercial manned platform and conduct real-world experiments in complex environments to validate the practicality of our algorithm. Moreover, we fully open-source our planning algorithm to further promote the development of vehicle motion planning.[1]

II. Related Work

A. Trajectory Generations for Vehicles

Sampling-based and search-based methods [11]–[24]are widely used for robot motion planning [25] due to the ease of incorporating user-defined objectives. Lattice-based planners [11]–[14] discretize the continuous state space into a lattice graph for planning. Safe Interval Path Planning (SIPP) and its variants [22]–[24] further decompose the temporal space into feasible time intervals for dynamic obstacle avoidance. Then, graph-search algorithms such as Dijkstra and A* are used to obtain the optimal trajectory in the graph. Ren et al. [23] extend SIPP to address multi-objective path planning by leveraging the concept of safe intervals from SIPP and multi-objective search techniques. Alabedeen et al. [24] effectively integrate kinodynamic constraints into SIPP, thereby improving its practicality and completeness. Probabilistic planners [15]–[21] represented by rapidly-exploring random tree (RRT) [15] obtain a feasible path by expanding a state tree rooted at the starting node. Despite the ability to avoid local minima in non-convex space, these typical methods require sampling in discrete robot state space to find a feasible trajectory, which confronts a dilemma between computation consumption and trajectory quality. To improve the alignment between trajectories and motion models, these methods often require sampling in higher-dimensional state spaces, which can lead to a combinatorial explosion. While these methods have theoretical optimality guarantees, achieving such guarantees demands high-resolution sampling and a significant number of samples, which imposes a substantial computational burden. The trade-off between time and trajectory quality presents a challenge for these methods to efficiently generate high-quality trajectories in complex environments.

To simplify the trajectory generation problem, some intuitive approaches [26]–[28] decouple the spatial shape and dynamics profile of the trajectory. Zhu et al. [26] propose convex elastic band smoothing (CES) algorithm which eliminates the non-convexity of the curvature constraint with fixed path lengths at each iteration, and transforms the original problem into a quadratically constrained quadratic program (QCQP). However, as presented in the work [28], the length consistency assumption does not always hold, which invalidates the curvature constraint and thus reduces the feasibility of the control. Based on CES framework, Zhou et al. [28] propose the dual-loop iterative anchoring path smoothing (DL-IAPS) algorithm to generate a smooth and safe path, where sequential convex optimization (SCP) is used to relax the curvature constraint. Whereas, the efficiency of this method relies heavily on the initial path obtained by hybridA* [29], which limits the application in complex environments.

Highly adaptable model predictive control (MPC) approaches formulate a trajectory planning problem as an optimal control problem (OCP) which is further discretized into a nonlinear programming (NLP) problem. These approaches [6]–[10, 30]–[34] optimize discrete states and control inputs, which can conveniently integrate dynamic and collision-free constraints. In the work [33], an iterative optimization framework is proposed to enhance the robustness to the initial corridor and improve trajectory quality. The work [34] develops an approach for automatic overtaking by establishing a multiphase constrained OCP and introducing a double-layer optimization framework. In the outer layer, an adaptive gradient-assisted particle swarm optimization algorithm is designed to generate near-optimal reference solutions, which are further optimized by the inner gradient-based optimization layer. Although the above MPC-based methods can easily accommodate the motion model of robots, since the trajectory is represented by discrete states, ensuring trajectory constraints in highly constrained environments requires increasing the density of states at the expense of degraded performance.

Several studies [35]–[39] demonstrate the potential of deep neural networks (DNNs) in motion planning. Qureshi et al. [35] propose a motion planning network that consists of an environment network (Enet) and a planning network (Pnet). The Enet extracts environment features, which are used by Pnet to output the next candidate state. Chai et al. [37] train multiple DNNs using a preplanned optimal parking trajectory dataset to learn the functional relationship between system state-control actions in parking maneuvers. A data aggregation method is also designed to enhance the DNN performance. This work [37] is further extended by designing a recurrent DNN capable of fully exploiting the inherent relationship between preoptimized system state and control pairs [39]. Additionally, two transfer model strategies are proposed to make the planner adaptable to different types of vehicles. While learning-based methods show effectiveness in certain tasks, their robustness to unknown scenarios may be limited. Moreover, these methods often require high-performance GPU parallel computing to achieve efficiency. Additionally, they typically lack interpretability and necessitate large amounts of data for model generalization, which constrains the practical applicability.

B. Obstacle Avoidance Formulation

Modeling the ego vehicle and other obstacles determines the complexity of collision-free constraints for trajectory generation problems. Li et al. [33] cover the ego vehicle with two circles along the centerline, and then the ego vehicle model is simplified to two points by inflating obstacles. In the work [33], obstacle avoidance is ensured by constraining these two points within a safe corridor. This approach essentially enjoys the property that the dimension of collision-free constraints is independent of the number of obstacles. However, such a point-based modeling method does not fully utilize the solution space and often generates overly conservative trajectories, especially in complex environments.

[1]https://github.com/ZJU-FAST-Lab/Dftpav

In the work [9], they considered the driving scenarios with arbitrarily placed obstacles and formulated the problem as a unified OCP for trajectory generation in unstructured environments. However, the collision-free constraint is nominally non-differentiable. Zhang et al. [40] assume that obstacles are convex and propose an optimization-based collision avoidance (OBCA) algorithm, which removes the integer variables used to model full-dimensional object collision avoidance [41]. They introduce dual variables to reformulate the distance between the robot and obstacles, and transform the collision-free constraint into a continuous differentiable form. Zhang et al. [31] integrate OBCA into the MPC problem of motion planning and propose H-OBCA, a hierarchical framework for trajectory planning in unstructured environments. Besides, a reasonable method is presented in the work [31] for warm-starting dual variables to speed up the optimization. However, the introduction of dual variables in OBCA-based methods [31, 32] increases the dimension of the problem, making it more challenging to solve. Besides, the number of dual variables is positively correlated with the number of obstacles. As obstacles increases, the problem dimension will rise rapidly, leading to unacceptable computational and memory costs.

In this paper, we focus on optimization-based trajectory planning for its theoretical guarantees and ability to accurately converge to local optimal solutions in continuous space. Without exploiting specific task features or artificially defined rules, our planner is sufficiently general and robust to adapt to various complex scenarios. Overall, instead of sampling the planning state, decoupling spatial and temporal space, discretizing the motion model, or relying heavily on environmental characteristics, our method incorporates spatial-temporal joint optimization to efficiently generate a high-quality trajectory with guaranteed full-dimensional obstacle avoidance. Our method enjoys low computational cost and high robustness against unstructured environments, while more detailed quantitative comparisons are presented in Sect. VII.

III. Spatial-Temporal Trajectory Planning

In this section, we present the spatial-temporal joint optimization formulation for trajectory planning. To construct the optimization formulation, we discuss the complete motion planning pipeline and introduce the differential flat model of car-like robots. Then, we show the formulation of the trajectory optimization problem considering human comfort, execution time, and feasibility constraints. Last but not least, we analyze the gradient propagation chain of the problem for subsequent numerical optimization.

A. Planning Pipeline

Our approach follows a hierarchical structure with a front-end whose main role is to provide an initial guess, and back-end optimization. We adopt the lightweight hybridA* algorithm to find a collision-free path that is further optimized by the proposed planner. In the implementation of the algorithm, the robot's state is defined by its position and orientation angle. Moreover, for each state to be expanded, we try using Reed Shepp Curve [42] to shoot the end state for the earlier termination of the search process. For complex driving tasks such as autonomous parking, the front-end output often contain both forward and back vehicle movements. By reasonably assuming the vehicle always reaches a complete stop at the gear shifting position, we parameterize the forward and backward segments of the trajectory as piece-wise polynomials, respectively, whose specific formulations are presented in Sect. III-C. Additionally, the motion direction of each segment is determined by the front-end output and prefixed before the back-end optimization process.

B. Differentially Flat Vehicle Model

We use the simplified kinematic bicycle model in the Cartesian coordinate frame to describe a four-wheel vehicle. Assuming that the car is front-wheel driven and steered with perfect rolling and no slipping, the model can be described as Fig. 2. The state vector is $\boldsymbol{x} = (p_x, p_y, \theta, v, a_t, a_n, \phi, \kappa)^{\mathrm{T}}$, where $\boldsymbol{p} = (p_x, p_y)^{\mathrm{T}}$ denotes the position at the center of the rear wheels, v is the longitudinal velocity w.r.t vehicle's body frame, a_t represents the longitude acceleration, a_n is the latitude acceleration, ϕ is the steering angle of the front wheels and κ is the instantaneous curvature of the trajectory. Thanks to the thorough study of the differentially flat car model [43], we choose the flat output as $\boldsymbol{\sigma} := (\sigma_x, \sigma_y)^{\mathrm{T}}$ with a physical meaning that $\boldsymbol{\sigma} = \boldsymbol{p}$ is the position centered on the rear wheel of the car. Other variable transformations except p_x, p_y can be expressed as:

$$v = \eta\sqrt{\dot{\sigma_x}^2 + \dot{\sigma_y}^2}, \tag{1a}$$

$$\theta = \arctan 2(\eta\dot{\sigma_y}, \eta\dot{\sigma_x}), \tag{1b}$$

$$a_t = \eta(\dot{\sigma_x}\ddot{\sigma_x} + \dot{\sigma_y}\ddot{\sigma_y})/\sqrt{\dot{\sigma_x}^2 + \dot{\sigma_y}^2}, \tag{1c}$$

$$a_n = \eta(\dot{\sigma_x}\ddot{\sigma_y} - \dot{\sigma_y}\ddot{\sigma_x})/\sqrt{\dot{\sigma_x}^2 + \dot{\sigma_y}^2}, \tag{1d}$$

$$\phi = \arctan\left(\eta(\dot{\sigma_x}\ddot{\sigma_y} - \dot{\sigma_y}\ddot{\sigma_x})L/(\dot{\sigma_x}^2 + \dot{\sigma_y}^2)^{\frac{3}{2}}\right), \tag{1e}$$

$$\kappa = \eta(\dot{\sigma_x}\ddot{\sigma_y} - \dot{\sigma_y}\ddot{\sigma_x})/(\dot{\sigma_x}^2 + \dot{\sigma_y}^2)^{\frac{3}{2}}. \tag{1f}$$

Consequently, with the natural differential flatness property, we can use the flat outputs and their finite derivatives to characterize arbitrary state quantities of the vehicle, which simplifies the trajectory planning and facilitates optimization. We define an additional variable $\eta \in \{-1, 1\}$ to characterize the motion direction of the vehicle, where $\eta = -1$ and $\eta = 1$ represent the backward and forward movements, respectively. We avoid singularities by fixing the velocity magnitude to a small, non-zero constant when the gear shifts. Both the gear shifting position and directional angle can be optimized, and their specific formulations are presented in Sect. VI.

C. Optimization Formulation

The i-th segment of the trajectory is formulated as a 2-dimensional and time-uniform M_i-piece polynomial with degree $N = 2s-1$, which is parameterized by the intermediate waypoints $\boldsymbol{q}_i = \left(\boldsymbol{q}_{i,1}, ..., \boldsymbol{q}_{i,M_i-1}\right) \in \mathbb{R}^{2\times(M_i-1)}$, the time interval for each piece $\delta T_i \in \mathbb{R}^+$, and the coefficient matrix

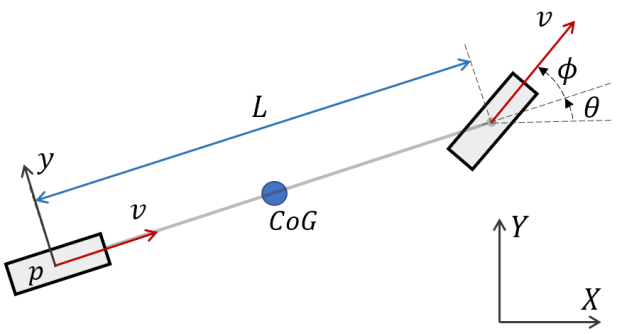


Fig. 2: The kinematic bicycle model.

$\boldsymbol{c}_i = \left(\boldsymbol{c}_{i,1}^{\mathrm{T}}, ..., \boldsymbol{c}_{i,M_i}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2M_i s \times 2}$. Then, the j-th piece of the i-th segment $\boldsymbol{\sigma}_{i,j}$ is written as:

$$\begin{aligned} \boldsymbol{\sigma}_{i,j}(t) &:= \boldsymbol{c}_{i,j}^{\mathrm{T}} \boldsymbol{\beta}(t), \\ \boldsymbol{\beta}(t) &:= \left(1, t, t^2, ..., t^N\right)^{\mathrm{T}}, \\ \forall t \in [0, \delta T_i], \forall i \in \{1, 2, ..., n\}, \forall j \in \{1, 2, 3, ..., M_i\}, \end{aligned} \tag{2}$$

where n is the number of trajectory segments and $\boldsymbol{\beta}(t)$ is a natural basis. The M_i-piece polynomial trajectory $\boldsymbol{\sigma}_i : [0, T_i]$ is obtained:

$$\begin{aligned} \boldsymbol{\sigma}_i(t) = \boldsymbol{\sigma}_{i,j}(t - (j-1) * \delta T_i), \\ \forall j \in \{1, 2, ..., M_i\}, t \in [(j-1) * \delta T_i, j * \delta T_i). \end{aligned} \tag{3}$$

Here, the total duration of the i-th segment of the trajectory is $T_i = M_i * \delta T_i$. Then, the complete trajectory representation $\boldsymbol{\sigma}(t) : [0, T_s]$ is formulated:

$$\begin{aligned} \boldsymbol{\sigma}(t) = \boldsymbol{\sigma}_i(t - \hat{T}_i), \\ \forall i \in \{1, 2, ..., n\}, t \in [\hat{T}_i, \hat{T}_{i+1}), \end{aligned} \tag{4}$$

where $T_s = \sum_{i=1}^{n} T_i$ is the duration of the whole trajectory, $\hat{T}_i = \sum_{\bar{i}=1}^{i-1} T_{\bar{i}}$ is the timestamp of the starting point of the i-th segment and $\hat{T}_1$ is set as 0. Moreover, we define a coefficient set $\mathbf{c} = \left(\boldsymbol{c}_1^{\mathrm{T}}, \boldsymbol{c}_2^{\mathrm{T}}, ..., \boldsymbol{c}_n^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{\left(\sum_{i=1}^{n} 2M_i s\right) \times 2}$ and a time set $\mathbf{T} = \left(T_1, T_2, ..., T_n\right)^{\mathrm{T}} \in \mathbb{R}^n$ for the subsequent derivation. With constraints for obstacles avoidance and dynamic feasibility, the minimal control effort problem involving time regularization can be expressed as a nonlinear constrained optimization:

$$\min_{\mathbf{c}, \mathbf{T}} J(\mathbf{c}, \mathbf{T}) = \int_0^{T_s} \boldsymbol{\mu}(t)^{\mathrm{T}} \mathbf{W} \boldsymbol{\mu}(t) dt + w_T T_s \tag{5a}$$

$$\text{s.t.} \boldsymbol{\mu}(t) = \boldsymbol{\sigma}^{[s]}(t), \forall t \in [0, T_s], \tag{5b}$$

$$\boldsymbol{\sigma}_0^{[s-1]}(0) = \bar{\boldsymbol{\sigma}}_0, \ \boldsymbol{\sigma}_n^{[s-1]}(T_n) = \bar{\boldsymbol{\sigma}}_f, \tag{5c}$$

$$\boldsymbol{\sigma}_i^{[s-1]}(T_i) = \boldsymbol{\sigma}_{i+1}^{[s-1]}(0) = \widetilde{\boldsymbol{\sigma}}_i, 1 \le i < n, \tag{5d}$$

$$\boldsymbol{\sigma}_{i,j}^{[\tilde{d}]}(\delta T_i) = \boldsymbol{\sigma}_{i,j+1}^{[\tilde{d}]}(0), 1 \le i \le n, 1 \le j < M_i, \tag{5e}$$

$$T_i > 0, 1 \le i \le n, \tag{5f}$$

$$\mathcal{G}_d(\boldsymbol{\sigma}(t), ..., \boldsymbol{\sigma}^{(s)}(t), t) \preceq 0, \quad \forall d \in \mathcal{D}, \forall t \in [0, T_s], \tag{5g}$$

where $\mathbf{W} \in \mathbb{R}^{2 \times 2}$ is a diagonal matrix to penalize control efforts. Eq. (5c) is the boundary condition, where $\bar{\boldsymbol{\sigma}}_0, \bar{\boldsymbol{\sigma}}_f \in \mathbb{R}^{2 \times s}$ are the user-specified initial and final states. $\widetilde{\boldsymbol{\sigma}}_i \in \mathbb{R}^{2 \times s}$ is the switching state at GSP. To enhance robustness, the position and the tangential direction of the motion curve are optimized, and the acceleration is fixed to zero to avoid over-jitter. Moreover, the specific formulation of the constraint Eq. (5d) and the corresponding optimization will be described in detail in Sect. VI-C. Eq. (5e) is the continuity constraint up to degree $\tilde{d}$. The second term $w_T T_s$ in the objective function is the time regularization term to restrict the total duration T_s, with a weight $w_T \in \mathbb{R}^+$. The constraint function at d is defined as $\mathcal{G}_d$. In our formulation, the set $\mathcal{D} = \{d : d = v, a_t, a_n, \kappa, \zeta, \Theta\}$ includes dynamic feasibility (v, a_t, a_n, κ), static and dynamic obstacle avoidance (ζ, Θ). Besides, s is chosen as 3, which means that the integration of jerk is minimized to ensure human comfort [44].

D. Gradient Derivation

Without loss of completeness, we first derive the analytic gradients of the objective function J w.r.t $\boldsymbol{c}_{i,j}$ and T_i:

$$\frac{\partial J}{\partial \boldsymbol{c}_{i,j}} = 2 \left(\int_0^{\delta T_i} \boldsymbol{\beta}^{(s)}(t) \boldsymbol{\beta}^{(s)}(t)^{\mathrm{T}} dt \right) \boldsymbol{c}_{i,j}, \tag{6}$$

$$\frac{\partial J}{\partial T_i} = \frac{1}{M_i} \sum_{j=1}^{M_i} \boldsymbol{c}_{i,j}^{\mathrm{T}} \boldsymbol{\beta}^{(s)}(\delta T_i) \boldsymbol{\beta}^{(s)}(\delta T_i)^{\mathrm{T}} \mathbf{c}_{i,j} + w_T. \tag{7}$$

The feasibility constraints Eq.(5g) imposed on the entire trajectory are equivalent to each piece of any piece-wise polynomial trajectory segment complying with these constraints:

$$\begin{aligned} &\mathcal{G}_d(\boldsymbol{\sigma}(t), ..., \boldsymbol{\sigma}^{(s)}(t), t) \preceq 0, \forall d \in \mathcal{D}, \forall t \in [0, T_s] \iff \\ &\mathcal{G}_d(\boldsymbol{\sigma}_{i,j}(\bar{t}), ..., \boldsymbol{\sigma}_{i,j}^{(s)}(\bar{t}), \hat{t}) \preceq 0, \forall d \in \mathcal{D}, \forall \bar{t} \in [0, \delta T_i], \\ &\quad \forall i \in \{1, 2, ..., n\}, \forall j \in \{1, 2, 3, ..., M_i\}, \end{aligned} \tag{8}$$

where $\bar{t}$ is the relative timestamp and $\hat{t} = \hat{T}_i + \delta T_i * (j-1) + \bar{t}$ is the absolute timestamp. Therefore, to approximate the continuous-time formula Eq. (8), we uniformly discretize each piece of the piece-wise polynomial into $\lambda \in \mathbb{N}_{>0}$ constraint points. Moreover, we ensure trajectory feasibility by imposing constraints on these constraint points. Then, the continuous-time formula Eq. (8) is transformed into a discrete form:

$$\begin{aligned} &\mathcal{G}_{d,i,j,k}(\boldsymbol{c}_{i,j}, \mathbf{T}) \preceq 0, \\ &\mathcal{G}_{d,i,j,k}(\boldsymbol{c}_{i,j}, \mathbf{T}) := \mathcal{G}_d(\boldsymbol{\sigma}_{i,j,k}, ..., \boldsymbol{\sigma}_{i,j,k}^{(s)}, \hat{t}), \\ &\boldsymbol{\sigma}_{i,j,k}^{(\bar{d})} := \boldsymbol{\sigma}_{i,j}^{(\bar{d})}(\bar{t}), \quad \forall \bar{d} \in \{0, 1, ..., s\}, \\ &\bar{t} = \frac{k T_i}{\lambda M_i}, \quad \hat{t} = \hat{T}_i + \left(\frac{j-1}{M_i} + \frac{k}{\lambda M_i} \right) T_i, \\ &\forall k \in \{0, 1, 2, ..., \lambda\}, \quad \forall d \in \mathcal{D}. \end{aligned} \tag{9}$$

Without loss of generality, we derive the gradient propagation at any constraint point based on the chain rule:

$$\frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \boldsymbol{c}_{i,j}} = \sum_{\bar{d}=0}^{s} \beta^{(\bar{d})}(\bar{t}) \left(\frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \boldsymbol{\sigma}_{i,j,k}^{(\bar{d})}} \right)^{\mathrm{T}}, \tag{10}$$

$$\frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \mathbf{T}} = \frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \bar{t}} \frac{\partial \bar{t}}{\partial \mathbf{T}} + \frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \hat{t}} \frac{\partial \hat{t}}{\partial \mathbf{T}}, \tag{11}$$

We further derive time-related gradient terms inside Eq. (11):

$$\frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \bar{t}} = \sum_{\bar{d}=0}^{s} \left(\boldsymbol{\sigma}_{i,j,k}^{(\bar{d}+1)}\right)^{\mathrm{T}} \frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \boldsymbol{\sigma}_{i,j,k}^{(\bar{d})}}, \tag{12}$$

$$\frac{\partial \bar{t}}{\partial \mathbf{T}} = \left(\mathbf{0}_{i-1}^{\mathrm{T}}, \frac{k}{\lambda M_i}, \mathbf{0}_{n-i}^{\mathrm{T}}\right)^{\mathrm{T}}, \tag{13}$$

$$\frac{\partial \hat{t}}{\partial \mathbf{T}} = \left(\mathbf{1}_{i-1}^{\mathrm{T}}, \frac{k}{\lambda M_i} + \frac{j-1}{M_i}, \mathbf{0}_{n-i}^{\mathrm{T}}\right)^{\mathrm{T}}, \tag{14}$$

where all the above gradient calculations w.r.t vectors follow the denominator layout. As a result, by substituting Eq.(12)-(14) to Eq.(11), we can obtain the gradients of $G_{d,i,j,k}$ w.r.t $\boldsymbol{c}_{i,j}$ and $\mathbf{T}$ once $\partial \mathcal{G}_{d,i,j,k}/\partial \boldsymbol{\sigma}_{i,j,k}^{(\bar{d})}$ and $\partial \mathcal{G}_{d,i,j,k}/\partial \hat{t}$ are specified. Besides, it is worth mentioning that constraint functions $G_{d,i,j,k}$ can be precisely expressed by some of the quantities in $\hat{t}$ and $\boldsymbol{\sigma}_{i,j,k}^{(\bar{d})}$, where $\bar{d} \in \{0, 1, 2..., s\}$. Therefore, the gradients w.r.t irrelevant variables are 0 without derivation. In subsequent sections, we present the specific formulation of the constraint functions $\mathcal{G}_{d\in\mathcal{D}}$ and derive gradients. For simplification, i, j, k and relative timestamp $\bar{t}$ are omitted in Sect. IV and Sect. V.

IV. Instantaneous State Constraints

In this section, we introduce the instantaneous state constraints for trajectory optimization, where these constraint functions are only related to the instantaneous states of the vehicle.

A. Dynamic Feasibility

1) Longitude Velocity Limit: For autonomous driving, the longitude velocity always needs to be limited within a reasonable range because of practical factors such as traffic rules, physical vehicle performance, and environmental uncertainty. Then, the constraint function of longitude velocity at a constraint point is defined as follows:

$$\mathcal{G}_v(\dot{\boldsymbol{\sigma}}) = \dot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}} - v_m^2. \tag{15}$$

where v_m is the magnitude of maximal longitude velocity. The gradient of $\mathcal{G}_v(\dot{\boldsymbol{\sigma}})$ is written as:

$$\frac{\partial \mathcal{G}_v}{\partial \dot{\boldsymbol{\sigma}}} = 2\dot{\boldsymbol{\sigma}}. \tag{16}$$

Accordingly, we can obtain the gradients of $\mathcal{G}_v$ by combining Eq.(16) and Eq.(10)-(14).

2) Acceleration Limit: The acceleration is always required to be limited to prevent skidding due to friction limits between the tire and the ground. From Eq. (1c)(1d), we define the constraint functions of longitude and latitude acceleration at a constraint point:

$$\mathcal{G}_{a_t}(\dot{\boldsymbol{\sigma}}, \ddot{\boldsymbol{\sigma}}) = \frac{(\ddot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}})^2}{\dot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}}} - a_{tm}^2, \tag{17}$$

$$\mathcal{G}_{a_n}(\dot{\boldsymbol{\sigma}}, \ddot{\boldsymbol{\sigma}}) = \frac{(\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}})^2}{\dot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}}} - a_{nm}^2, \tag{18}$$

where a_{tm} and a_{nm} are the maximal longitude and latitude acceleration and $\mathbf{B} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ is an auxiliary antisymmetric matrix. The gradients of $\mathcal{G}_{a_t}(\dot{\boldsymbol{\sigma}}, \ddot{\boldsymbol{\sigma}})$ and $\mathcal{G}_{a_n}(\dot{\boldsymbol{\sigma}}, \ddot{\boldsymbol{\sigma}})$ are derived as :

$$\frac{\partial \mathcal{G}_{a_t}}{\partial \dot{\boldsymbol{\sigma}}} = 2\frac{\ddot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^2} \ddot{\boldsymbol{\sigma}} - 2\left(\frac{\ddot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^2}\right)^2 \dot{\boldsymbol{\sigma}}, \tag{19}$$

$$\frac{\partial \mathcal{G}_{a_n}}{\partial \dot{\boldsymbol{\sigma}}} = 2\frac{\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^2} \mathbf{B}^T \ddot{\boldsymbol{\sigma}} - 2\left(\frac{\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^2}\right)^2 \dot{\boldsymbol{\sigma}}, \tag{20}$$

$$\frac{\partial \mathcal{G}_{a_t}}{\partial \ddot{\boldsymbol{\sigma}}} = 2\frac{\ddot{\boldsymbol{\sigma}}^T \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^2} \dot{\boldsymbol{\sigma}}, \frac{\partial \mathcal{G}_{a_n}}{\partial \ddot{\boldsymbol{\sigma}}} = 2\frac{\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^2} \mathbf{B} \dot{\boldsymbol{\sigma}}. \tag{21}$$

3) Front Steer Angle Limit: The front steer angle needs to be limited to ensure the nonholonomic dynamic feasibility of the vehicle. Due to the monotonicity of the tangent function, we restrict the front steer angle by limiting the curvature $\kappa = \tan\phi/L$ in $[-\tan\phi_m/L, \tan\phi_m/L] := [-\kappa_m, \kappa_m]$, where ϕ_m is the preset maximum steer angle and κ_m is the corresponding maximum curvature. Then, the nonholonomic dynamic constraint function $\mathcal{G}_\kappa$ is expressed as:

$$\mathcal{G}_\kappa(\dot{\boldsymbol{\sigma}}, \ddot{\boldsymbol{\sigma}}) = (\frac{\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^3})^2 - \kappa_m^2. \tag{22}$$

Furthermore, we derive the gradients w.r.t $\dot{\boldsymbol{\sigma}}$ and $\ddot{\boldsymbol{\sigma}}$:

$$\frac{\partial \mathcal{G}_\kappa}{\partial \dot{\boldsymbol{\sigma}}} = 2(\frac{\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^3})(\frac{\mathbf{B}^T \ddot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^3} - 3\frac{\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^5} \dot{\boldsymbol{\sigma}}), \tag{23}$$

$$\frac{\partial \mathcal{G}_\kappa}{\partial \ddot{\boldsymbol{\sigma}}} = 2(\ddot{\boldsymbol{\sigma}}^T \mathbf{B} \dot{\boldsymbol{\sigma}}) \frac{\mathbf{B} \dot{\boldsymbol{\sigma}}}{||\dot{\boldsymbol{\sigma}}||_2^6}. \tag{24}$$

B. Static Obstacle Avoidance

In this subsection, we analytically present static collision-free constraints that are efficiently computable based on the geometric representation of the free space in the environment. We first decompose the semantic environment and extract the collision-free space to construct a driving corridor consisting of a series of convex polygons. Then, we derive the necessary and sufficient condition for enforcing the full vehicle shape in the driving corridor, which is used to construct static no-collision constraints. Before specific derivation, we introduce the pipeline of the constraint modeling. We first discretize the collision-free path generated by the front end into sampling points whose number is the same as the number of constraint points in the back-end optimization. Then, combined with the environmental information, we generate a free convex polygon based on the sampling point by the method [45] or directly expanding in each defined direction. As a result, the entire trajectory is guaranteed to be collision-free by confining the full vehicle shape at each constraint point to the corresponding convex polygon, as shown in Fig. 3. We use a convex polygon to enclose the full shape of the ego vehicle which is defined as $\mathbb{E}$. Moreover, we define the vertice set $\mathcal{E}$ of the convex polygon as:

$$\mathcal{E} = \{\boldsymbol{v}_e \in \mathbb{R}^2 : \boldsymbol{v}_e = \boldsymbol{\sigma} + \mathbf{R}\boldsymbol{l}_e, e = 1, 2, ..., n_e\}, \tag{25}$$

where $\mathbf{R}$ is the rotation matrix from the body to the world frame, transformed into flat outputs as:

$$\mathbf{R} = \frac{\eta}{\|\dot{\boldsymbol{\sigma}}\|_2} \left(\dot{\boldsymbol{\sigma}}, \mathbf{B}\dot{\boldsymbol{\sigma}}\right). \tag{26}$$

Here, n_e is the number of vertexes, and $\boldsymbol{l}_e$ is the coordinate of the e-th vertex in the body frame. $\eta \in \{-1, 1\}$ is a prefixed auxiliary variable to indicate the motion direction of the segment of the trajectory. Note n_e and $\boldsymbol{l}_e$ are also constant once the vehicle shape is identified. The H-representation [46] of each convex polygon $\mathcal{P}^H$ in the driving corridor is obtained:

$$\begin{aligned} \mathcal{P}^H &= \{\boldsymbol{q} \in \mathbb{R}^2 : \boldsymbol{A}\boldsymbol{q} \leq \boldsymbol{b}\}, \\ \boldsymbol{A} &= (\boldsymbol{A}_1, ..., \boldsymbol{A}_z, ..., \boldsymbol{A}_{n_z})^{\mathrm{T}} \in \mathbb{R}^{n_z \times 2}, \\ \boldsymbol{b} &= (b_1, ..., b_z, ..., b_{n_z})^{\mathrm{T}} \in \mathbb{R}^{n_z}, \end{aligned} \tag{27}$$

where n_z is the number of hyperplanes, $\boldsymbol{A}_z \in \mathbb{R}^2$ and $b_z \in \mathbb{R}$ are the descriptors of a hyperplane, which can be determined by a point on the hyperplane and the normal vector. Additionally, once the driving corridor is generated, the hyperplane descriptors $\boldsymbol{A}_z$ and b_z are also completely determined. A sufficient and necessary condition for containing the full shape of the vehicle in a convex polygon is that each vertex of the vehicle shape is contained in the convex polygon:

$$\begin{aligned} &\mathbb{E} \subseteq \mathcal{P}^H \iff \\ &\quad \boldsymbol{\sigma} + \mathbf{R}\boldsymbol{l}_e \subseteq \mathcal{P}^H \quad \forall e \in \{1, 2, ..., n_e\}. \end{aligned} \tag{28}$$

A sufficient and necessary condition for containing a vertex in a convex polygon is that the vertex is on the inner side of each hyperplane:

$$\begin{aligned} &\boldsymbol{\sigma} + \mathbf{R}\boldsymbol{l}_e \subseteq \mathcal{P}^H \iff \\ &\quad \boldsymbol{A}_z^{\mathrm{T}} (\boldsymbol{\sigma} + \mathbf{R}\boldsymbol{l}_e) \leq b_z \quad \forall z \in \{1, 2, ..., n_z\}. \end{aligned} \tag{29}$$

Therefore, the spatial constraint function at a constraint point is $\mathcal{G}_\zeta = (\mathcal{G}_{\zeta_{1,1}}, ..., \mathcal{G}_{\zeta_{e,z}}, ..., \mathcal{G}_{\zeta_{n_e,n_z}})^{\mathrm{T}} \in \mathbb{R}^{n_e n_z}$, with $n_e n_z$ linear constraint penalty about vertices of the ego vehicle, which is defined as:

$$\mathcal{G}_{\zeta_{e,z}}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}) = \boldsymbol{A}_z^{\mathrm{T}}(\boldsymbol{\sigma} + \mathbf{R}\boldsymbol{l}_e) - b_z. \tag{30}$$

Before further derivation, we define an auxiliary expression $\mathcal{F}(\boldsymbol{l}) : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ to simplify the form:

$$\mathcal{F}(\boldsymbol{l}) = \frac{\eta\,(\boldsymbol{l}, \mathbf{B}\boldsymbol{l})^{\mathrm{T}}}{\|\dot{\boldsymbol{\sigma}}\|_2} - \frac{\dot{\boldsymbol{\sigma}}\,(\mathbf{R}\boldsymbol{l})^{\mathrm{T}}}{\|\dot{\boldsymbol{\sigma}}\|_2^2}. \tag{31}$$

The gradients of the constraint function w.r.t $\boldsymbol{\sigma}$ and $\dot{\boldsymbol{\sigma}}$ are:

$$\frac{\partial \mathcal{G}_{\zeta_{e,z}}}{\partial \boldsymbol{\sigma}} = \boldsymbol{A}_z, \tag{32}$$

$$\frac{\partial \mathcal{G}_{\zeta_{e,z}}}{\partial \dot{\boldsymbol{\sigma}}} = \mathcal{F}(\boldsymbol{l}_e)\boldsymbol{A}_z. \tag{33}$$

The gradients w.r.t the polynomial coefficients and durations can also be calculated by propagating equations Eq.(10)-(14).

V. Dynamic Obstacle Avoidance

Dynamic collision-free property is guaranteed by ensuring that the minimum distance between the ego vehicle and obstacle convex polygons at each moment of the trajectory is greater than the collision-free threshold. To increase readability, we introduce helpful priors for evaluating the signed distance between convex polygons. Then, we elaborate on the dynamic avoidance constraint function which is further relaxed into a continuously differentiable form.

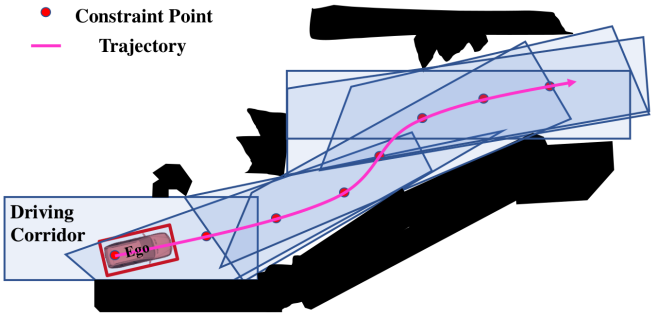


Fig. 3: Visualization of the driving corridor. The convex polygon contains the full shape of the ego vehicle at any constraint point. With a proper trajectory resolution, we can practically guarantee the static no-collision property.

A. Preliminaries on Distance Representations

1) Signed Distances for Rigid Objects: We consider two convex polytopes $\mathbb{E}, \mathbb{O}$ bounded by the intersection of halfspaces, as $\mathbb{E} = \cap_{e=1}^{K_e} \mathcal{P}_e$, $\mathbb{O} = \cap_{o=1}^{K_o} \mathcal{P}_o$. The usual convex collision avoidance method penalizes the signed distance between two sets. The distance is defined with the minimal translation $\mathcal{T}$ as:

$$\mathrm{dist}(\mathbb{E}, \mathbb{O}) = \min_{\mathcal{T}} \{\|\mathcal{T}\| : (\mathbb{E} + \mathcal{T}) \cap O \neq \emptyset\}. \tag{34}$$

When overlapping, $\mathrm{dist} = 0$ holds, which is insufficient to obtain gradient directions to separate them. The penetration depth can be combined to solve this issue:

$$\mathrm{pen}(\mathbb{E}, \mathbb{O}) = \min_{\mathcal{T}} \{\|\mathcal{T}\| : (\mathbb{E} + \mathcal{T}) \cap O = \emptyset\}. \tag{35}$$

Hence, we can get the signed distance:

$$\mathrm{sd}(\mathbb{E}, \mathbb{O}) := \mathrm{dist}(\mathbb{E}, \mathbb{O}) - \mathrm{pen}(\mathbb{E}, \mathbb{O}). \tag{36}$$

Computing the signed distance requires solving minimum optimization problems Eq. (34, 35), which is unsuitable for embedding into our trajectory optimization problem. We will discuss a Minkowski difference-based algorithm for the approximate efficient computation of signed distances in the next section.

2) Approximation Distances: We follow the definition of Minkowski difference used in GJK algorithm [47]. Considering the general case where $A, B \in \mathbb{R}^n$ are two sets, the Minkowski Difference is defined by:

$$A - B = \{a - b \in \mathbb{R}^n : a \in A, b \in B\}. \tag{37}$$

Based on the core property [48] extensively used for collision checking:

$$\mathrm{sd}(\mathbb{E}, \mathbb{O}) = \mathrm{sd}(\mathbf{0}, \mathbb{O} - \mathbb{E}). \tag{38}$$

The problem of computing signed distances between two sets can be reduced to the distance between the origin point $\mathbf{0}$ to the set $\mathbb{O} - \mathbb{E}$. A concise formulation to bound signed distances is proposed in [49], defined as the maximum signed distance from the origin to the Minkowski difference between

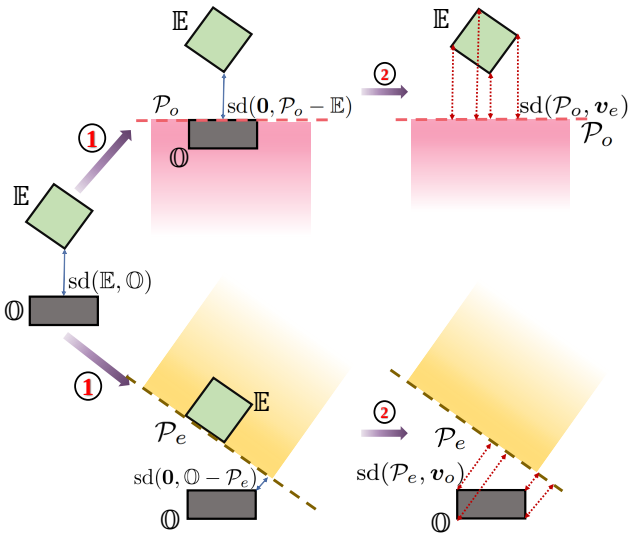


Fig. 4: Illustration of computing the lower bound of the signed distance between convex sets $\mathbb{E}$ and $\mathbb{O}$, where $\mathcal{P}_e$ and $\mathcal{P}_o$ refer to any hyperplane of $\mathbb{E}$ and $\mathbb{O}$. Besides, we define $\boldsymbol{v}_e$ and $\boldsymbol{v}_o$ as any vertex of $\mathbb{E}$ and $\mathbb{O}$, respectively. The computation process follows the above two-stage structure. We first use the maximum of the signed distances between convex sets and hyperplanes to approximate $\mathrm{sd}(\mathbb{E}, \mathbb{O})$. Then, due to the convexity of $\mathbb{E}$ and $\mathbb{O}$, $\mathrm{sd}(\mathbf{0}, \mathcal{P}_o - \mathbb{E})$ and $\mathrm{sd}(\mathbf{0}, \mathbb{O} - \mathcal{P}_e)$ are converted to point-to-hyperplane distances $\mathrm{sd}(\mathcal{P}_o, \boldsymbol{v}_e)$ and $\mathrm{sd}(\mathcal{P}_e, \boldsymbol{v}_o)$ which can be analytically calculated.

a polygon and each hyperplane:

$$\max_{\mathcal{P}_e, \mathcal{P}_o} \{\mathrm{sd}(\mathbf{0}, \mathbb{O} - \mathcal{P}_e), \mathrm{sd}(\mathbf{0}, \mathcal{P}_o - \mathbb{E})\} \leq \mathrm{sd}(\mathbb{E}, \mathbb{O}), \quad \forall e = \{1, ..., K_e\}, \forall o = \{1, ..., K_o\}. \tag{39}$$

By extending the Minkowski difference to the case between a polygon and each hyperplane, we have

$$\begin{aligned} \mathbb{O} - \mathcal{P}_e &= \{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} + \boldsymbol{y} \in \mathbb{O}, \boldsymbol{y} \in \mathcal{P}_e\}, \\ &= \{\boldsymbol{p} \in \mathbb{R}^n : (\mathbf{H}^e)^{\mathrm{T}} \boldsymbol{p} \geq -\mathrm{h}^e + (\mathbf{H}^e)^{\mathrm{T}} \boldsymbol{u},\ \boldsymbol{u} \in \mathbb{O}\}. \end{aligned} \tag{40}$$

with $\mathcal{P}_e = \{\boldsymbol{y} \in \mathbb{R}^n : (\mathbf{H}^e)^{\mathrm{T}} \boldsymbol{y} \leq \mathrm{h}^e\}$ as a hyperplane of the set $\mathbb{E}$. Similarly, we obtain

$$\mathcal{P}_o - \mathbb{E} = \{\boldsymbol{p} \in \mathbb{R}^n : (\mathbf{G}^o)^{\mathrm{T}} \boldsymbol{p} \leq \mathrm{g}^o - (\mathbf{G}^o)^{\mathrm{T}} \boldsymbol{y},\ \boldsymbol{y} \in \mathbb{E}\}. \tag{41}$$

The signed distance can be computed as:

$$\mathrm{sd}(\mathbf{0}, \mathbb{O} - \mathcal{P}_e) = \frac{1}{\|\mathbf{H}^e\|_2} (-\mathrm{h}^e + \min_{\boldsymbol{u}} (\mathbf{H}^e)^{\mathrm{T}} \boldsymbol{u}), \tag{42}$$

$$\mathrm{sd}(\mathbf{0}, \mathcal{P}_o - \mathbb{E}) = \frac{1}{\|\mathbf{G}^o\|_2} (-\mathrm{g}^o + \min_{\boldsymbol{y}} (\mathbf{G}^o)^{\mathrm{T}} \boldsymbol{y}). \tag{43}$$

The physical meaning of this formulation is that the signed distance of two sets is bounded by a maximum signed distance of one set to each hyperplane and vice versa. For convex polytopes, we only need to check each vertex to get the minimum value and then the lower bound can be analytically calculated for optimization, as shown in Fig. 4.

B. Constraint for Dynamic Avoidance

In this section, we present the specific form of the dynamic obstacle avoidance constraint. In practical autonomous driving scenarios, motion trajectories of dynamic objects are typically predicted by perception modules, which are outside the scope of this work. Therefore, in this paper, motion trajectories of dynamic obstacles are assumed to be known and inputted to the trajectory planner.

We apply discrete obstacle avoidance constraints on the constraint points along the trajectory regarding the trajectories of other obstacles at the same time stamp. Therefore, the dynamic collision-free constraint is defined as $\mathcal{G}_\Theta(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}, \hat{t}) = \{\mathcal{G}_{\Theta_1}, ..., \mathcal{G}_{\Theta_u}, ..., \mathcal{G}_{\Theta_{N_u}}\}^{\mathrm{T}} \in \mathbb{R}^{N_u}$ where N_u is the number of dynamic obstacles. The dynamic avoidance constraint function with the u-th moving object at a constraint point is defined as:

$$\mathcal{G}_{\Theta_u}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}, \hat{t}) = d_m - U(\mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}), \mathbb{O}_u(\hat{t}))), \tag{44}$$

where d_{m} is the minimum collision-free distance (collision-free margin) and $U(\mathbb{E}(t), \mathbb{O}_u(\hat{t}))$ is the distance between the ego vehicle and the moving obstacle. With the lower approximation of the signed distance between two convex objects (Sect. V-A2), we have:

$$\mathrm{sd}\left(\mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}), \mathbb{O}_u(\hat{t})\right) \geq \mathrm{lb}_{\mathrm{sd}}\left(\mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}), \mathbb{O}_u(\hat{t})\right). \tag{45}$$

The lower bound $\mathrm{lb}_{\mathrm{sd}}$ is paraphrased by Eq. (39) as:

$$\begin{aligned} &\mathrm{lb}_{\mathrm{sd}}\left(\mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}), \mathbb{O}_u(\hat{t})\right) = \\ &\max_{\mathcal{P}_e, \mathcal{P}_o^u} \left\{\mathrm{sd}(\mathbf{0}, \mathbb{O}_u(\hat{t}) - \mathcal{P}_e(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}})), \mathrm{sd}(\mathbf{0}, \mathcal{P}_o^u(\hat{t}) - \mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}))\right\}, \\ &\quad e \in \{1, ..., n_e\}, o \in \{1, ..., n_u\}, \end{aligned} \tag{46}$$

where n_u is the number of hyperplanes of the u-th moving obstacle and $\mathcal{P}_o^u$ is the hyperplane. Since the motion planning is on a two-dimensional plane, a hyperplane of a convex set degenerates into a straight line determined by two vertices. Then, the hyperplane descriptors of the ego vehicle can be determined as follows:

$$\mathbf{H}^e = \frac{\mathbf{B}\left(\boldsymbol{v}_{e+1} - \boldsymbol{v}_e\right)}{||\boldsymbol{v}_{e+1} - \boldsymbol{v}_e||_2}, \tag{47}$$

$$\mathrm{h}^e = (\mathbf{H}^e)^{\mathrm{T}} \boldsymbol{v}_e, e \in \{1, ..., n_e\}, \tag{48}$$

where $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_e, ..., \boldsymbol{v}_{n_e+1}\}$ are the vertices arranged clockwise, defined in Sec.IV-B, and $\boldsymbol{v}_{n_e+1} = \boldsymbol{v}_1$. Due to the convexity of the model, the distance between the moving obstacle and the hyperplane is converted into the minimum distance between vertices and the hyperplane, thus simplifying Eq.(42):

$$\begin{aligned} \mathrm{sd}(\mathbf{0}, \mathbb{O}_u - \mathcal{P}_e) &= \frac{1}{\|\mathbf{H}^e\|_2} (-\mathrm{h}^e + \min_{\boldsymbol{o}} (\mathbf{H}^e)^{\mathrm{T}} \boldsymbol{v}_o^u), \\ \boldsymbol{v}_o^u &= \boldsymbol{w}_u + \mathbf{R}_u \boldsymbol{l}_o^u, \quad o \in \{1, ..., n_u\}, \end{aligned} \tag{49}$$

where $\boldsymbol{v}_o^u$ is any vertex of the moving obstacle $\mathbb{O}_u$. $\boldsymbol{w}_u$ and $\mathbf{R}_u$ are the origin and rotation matrix of the obstacle body coordinate system, respectively. l_o^u is the translation vector determined in advance. Before further derivation, we define

an auxiliary expression $\mathcal{H}(\widetilde{\mathbf{R}}, \Delta\boldsymbol{l}) : (\mathbb{R}^{2\times2}, \mathbb{R}^2) \to \mathbb{R}^2$:

$$\mathcal{H}(\widetilde{\mathbf{R}}, \Delta\boldsymbol{l}) = \frac{\mathbf{B}\widetilde{\mathbf{R}}\Delta\boldsymbol{l}}{\|\Delta\boldsymbol{l}\|_2}. \tag{50}$$

By combining Eq.(25)(47-49), the signed distance can be analytically calculated:

$$\mathrm{sd}(\mathbf{0}, \mathbb{O}_u - \mathcal{P}_e) = \min_o \mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e)^{\mathrm{T}}(\boldsymbol{v}_o^u - \boldsymbol{v}_e), \tag{51}$$

where $\Delta\boldsymbol{l}_e = \boldsymbol{l}_{e+1} - \boldsymbol{l}_e$, and the physical meaning of $\mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e)$ is the normal vector outwards of the plane $\mathcal{P}_e$. Similarly, the signed distance between $\mathcal{P}_o^u$ and $\mathbb{E}$ can also be obtained:

$$\mathrm{sd}(\mathbf{0}, \mathcal{P}_o^u - \mathbb{E}) = \min_e \mathcal{H}(\mathbf{R}_u, \Delta\boldsymbol{l}_o^u)^{\mathrm{T}}\left(\boldsymbol{v}_e - \boldsymbol{v}_o^u\right). \tag{52}$$

Finally, we substitute Eq.(51)(52) into Eq.(46) to obtain the analytical expression of $\mathrm{lb}_{\mathrm{sd}}$:

$$\begin{aligned}
&\mathrm{lb}_{\mathrm{sd}}\left(\mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}), \mathbb{O}_u(\hat{t})\right) = \\
&\max\{\max_e \min_o \mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e)^{\mathrm{T}}(\boldsymbol{v}_o^u(\hat{t}) - \boldsymbol{v}_e), \\
&\max_o \min_e \mathcal{H}(\mathbf{R}_u(\hat{t}), \Delta\boldsymbol{l}_o^u)^{\mathrm{T}}\left(\boldsymbol{v}_e - \boldsymbol{v}_o^u(\hat{t})\right)\}, \\
&\quad e \in \{1, ..., n_e\}, o \in \{1, ..., n_u\}.
\end{aligned} \tag{53}$$

To smooth the maximum and minimum operations, a widely adopted log-sum-exp function is applied, defined as follows to approximate the vector-max(min) function:

$$\mathrm{lse}_\alpha(\boldsymbol{\gamma}) = \alpha^{-1}\log\left(\sum_{\omega=1}^{\Omega}\exp\left(\alpha r_\omega\right)\right), \tag{54}$$

where r_ω is a element of the vector $\boldsymbol{\gamma} = \{r_1, ..., r_\omega, ..., r_\Omega\}^{\mathrm{T}} \in \mathbb{R}^{\Omega>0}$. if $\alpha > 0$ then it approximately gets the maximum value in $\boldsymbol{\gamma}$, or $\alpha < 0$ selects the minimum term. We smooth the discrete function by the log-sum-exp function with the advantage that the gradient of the log-sum-exp function is exactly the softmax(min) function. Moreover, the approximation error of the log-sum-exp is lower bounded by:

$$\mathrm{lse}_{\alpha>0}(\boldsymbol{\gamma}) \geq \max\{\boldsymbol{\gamma}\} \geq \mathrm{lse}_{\alpha>0}(\boldsymbol{\gamma}) - \frac{\log(\Omega)}{\alpha}. \tag{55}$$

Hence, we can formulate the distance function:

$$\begin{aligned}
U(\mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}), \mathbb{O}_u(\hat{t})) &= \mathrm{lse}_{\alpha>0}(\mathbf{d}) - \frac{\log(n_e + n_u)}{\alpha}, \\
\mathbf{d} &= \left(\boldsymbol{d}_U{}^{\mathrm{T}}, \boldsymbol{d}_E{}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{n_e+n_u}, \\
\boldsymbol{d}_U &= \left(d_U^1, ..., d_U^e, ...d_U^{n_e}\right)^{\mathrm{T}} \in \mathbb{R}^{n_e}, \\
\boldsymbol{d}_E &= \left(d_E^1, ..., d_E^o, ...d_E^{n_u}\right)^{\mathrm{T}} \in \mathbb{R}^{n_u},
\end{aligned} \tag{56}$$

where $d_U^e = \mathrm{sd}(\mathbf{0}, \mathbb{O}_u(\hat{t}) - \mathcal{P}_e(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}))$ and $d_E^o = \mathrm{sd}(\mathbf{0}, \mathcal{P}_o^u(\hat{t}) - \mathbb{E}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}))$ are defined to simplify the formulation. Similarly, the minimum operation in d_U^e and d_E^o can by approximated by the log-sum-exp function with $\alpha < 0$. Combined with Eq.(51)(52) We transform the distance into the flat-output space as:

$$d_U^e = \mathrm{lse}_{\alpha<0}\left(\left(d_{U_1}^e, ..., d_{U_o}^e, ..., d_{U_{n_u}}^e\right)^{\mathrm{T}}\right) + \widetilde{d}_U^e, \tag{57}$$

$$d_E^o = \mathrm{lse}_{\alpha<0}\left(\left(d_{E_1}^o, ..., d_{E_e}^o, ..., d_{E_{n_e}}^o\right)^{\mathrm{T}}\right) + \widetilde{d}_E^o, \tag{58}$$

$$d_{U_o}^e = \mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e)^{\mathrm{T}}\mathbf{R}_u(\hat{t})\boldsymbol{l}_o^u, \widetilde{d}_U^e = \mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e)^{\mathrm{T}}(\boldsymbol{w}_u(\hat{t}) - \boldsymbol{v}_e),$$
$$d_{E_e}^o = \mathcal{H}(\mathbf{R}_u(\hat{t}), \Delta\boldsymbol{l}_o^u)^{\mathrm{T}}\mathbf{R}\boldsymbol{l}_e, \widetilde{d}_E^o = \mathcal{H}(\mathbf{R}_u(\hat{t}), \Delta\boldsymbol{l}_o^u)^{\mathrm{T}}(\boldsymbol{\sigma} - \boldsymbol{v}_o^u(\hat{t})).$$

By substituting Eq.(57,58) into Eq.(56) and then into Eq.(44), $\mathcal{G}_{\Theta_u}(\boldsymbol{\sigma}, \dot{\boldsymbol{\sigma}}, \hat{t})$ is transformed into a continuously differentiable function that can be analytically expressed by flat outputs. Based on the chain rule, we calculate the gradients:

$$\begin{aligned}
\frac{\partial\mathcal{G}_{\Theta_u}}{\partial\boldsymbol{\sigma}} &= -\sum_{e=1}^{n_e}\mathrm{lse}'_{\alpha>0}(d_U^e)\frac{\partial\widetilde{d}_U^e}{\partial\boldsymbol{\sigma}} - \sum_{o=1}^{n_u}\mathrm{lse}'_{\alpha>0}(d_E^o)\frac{\partial\widetilde{d}_E^o}{\partial\boldsymbol{\sigma}}, \\
\frac{\partial\widetilde{d}_U^e}{\partial\boldsymbol{\sigma}} &= -\mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e), \quad \frac{\partial\widetilde{d}_E^o}{\partial\boldsymbol{\sigma}} = \mathcal{H}(\mathbf{R}_u(\hat{t}), \Delta\boldsymbol{l}_o^u),
\end{aligned} \tag{59}$$

where $\mathrm{lse}' : \mathbb{R} \to \mathbb{R}$ is the gradient of the log-sum-exp function. In a similar way, we derive the gradients w.r.t $\hat{t}$ and $\dot{\boldsymbol{\sigma}}$:

$$\begin{aligned}
\frac{\partial\mathcal{G}_{\Theta_u}}{\partial\hat{t}} =& -\sum_{e=1}^{n_e}\mathrm{lse}'_{\alpha>0}(d_U^e)\left(\sum_{o=1}^{n_u}\mathrm{lse}'_{\alpha<0}(d_{U_o}^e)\frac{\partial d_{U_o}^e}{\partial\hat{t}} + \frac{\partial\widetilde{d}_U^e}{\partial\hat{t}}\right) \\
&-\sum_{o=1}^{n_u}\mathrm{lse}'_{\alpha>0}(d_E^o)\left(\sum_{e=1}^{n_e}\mathrm{lse}'_{\alpha<0}(d_{E_e}^o)\frac{\partial d_{E_e}^o}{\partial\hat{t}} + \frac{\partial\widetilde{d}_E^o}{\partial\hat{t}}\right), \\
\frac{\partial\mathcal{G}_{\Theta_u}}{\partial\dot{\boldsymbol{\sigma}}} =& -\sum_{e=1}^{n_e}\mathrm{lse}'_{\alpha>0}(d_U^e)\left(\sum_{o=1}^{n_u}\mathrm{lse}'_{\alpha<0}(d_{U_o}^e)\frac{\partial d_{U_o}^e}{\partial\dot{\boldsymbol{\sigma}}} + \frac{\partial\widetilde{d}_U^e}{\partial\dot{\boldsymbol{\sigma}}}\right) \\
&-\sum_{o=1}^{n_u}\mathrm{lse}'_{\alpha>0}(d_E^o)\sum_{e=1}^{n_e}\mathrm{lse}'_{\alpha<0}(d_{E_e}^o)\frac{\partial d_{E_e}^o}{\partial\dot{\boldsymbol{\sigma}}}.
\end{aligned} \tag{60}$$

From Eq.(60), we can obtain the gradients of the dynamic collision-free constraint once the gradients of auxiliary distances are determined. Next, we derive the gradients w.r.t $\dot{\boldsymbol{\sigma}}$:

$$\begin{aligned}
\frac{\partial d_{U_o}^e}{\partial\dot{\boldsymbol{\sigma}}} &= \frac{\mathcal{F}(\Delta\boldsymbol{l}_e)\mathbf{B}^{\mathrm{T}}\mathbf{R}_u(\hat{t})\boldsymbol{l}_o^u}{\|\Delta\boldsymbol{l}_e\|_2}, \\
\frac{\partial\widetilde{d}_U^e}{\partial\dot{\boldsymbol{\sigma}}} &= \frac{\mathcal{F}(\Delta\boldsymbol{l}_e)\mathbf{B}^{\mathrm{T}}(\boldsymbol{w}_u(\hat{t}) - \boldsymbol{v}_e) + \mathcal{F}(\boldsymbol{l}_e)\mathbf{B}^{\mathrm{T}}\mathbf{R}\Delta\boldsymbol{l}_e}{\|\Delta\boldsymbol{l}_e\|_2}, \\
\frac{\partial d_{E_e}^o}{\partial\dot{\boldsymbol{\sigma}}} &= \mathcal{F}(\boldsymbol{l}_e)\mathcal{H}(\mathbf{R}_u(\hat{t}), \Delta\boldsymbol{l}_o^u).
\end{aligned} \tag{61}$$

Then, the gradients w.r.t abstract timestamp $\hat{t}$ are also derived as follows:

$$\begin{aligned}
\frac{\partial d_{U_o}^e}{\partial\hat{t}} &= \left(\dot{\mathbf{R}}_u(\hat{t})\boldsymbol{l}_o^u\right)^{\mathrm{T}}\mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e), \\
\frac{\partial\widetilde{d}_U^e}{\partial\hat{t}} &= \left(\dot{\boldsymbol{w}}_u(\hat{t})\right)^{\mathrm{T}}\mathcal{H}(\mathbf{R}, \Delta\boldsymbol{l}_e), \\
\frac{\partial d_{E_e}^o}{\partial\hat{t}} &= \left(\mathbf{R}\boldsymbol{l}_e\right)^{\mathrm{T}}\mathcal{H}(\dot{\mathbf{R}}_u(\hat{t}), \Delta\boldsymbol{l}_o^u), \\
\frac{\partial\widetilde{d}_E^o}{\partial\hat{t}} &= \mathcal{H}(\dot{\mathbf{R}}_u(\hat{t}), \Delta\boldsymbol{l}_o^u)^{\mathrm{T}}(\boldsymbol{\sigma} - \boldsymbol{w}_u(\hat{t}) - \mathbf{R}_u(\hat{t})\boldsymbol{l}_o^u) \\
&\quad - \mathcal{H}(\mathbf{R}_u(\hat{t}), \Delta\boldsymbol{l}_o^u)^{\mathrm{T}}(\dot{\boldsymbol{w}}_u(\hat{t}) + \dot{\mathbf{R}}_u(\hat{t})\boldsymbol{l}_o^u),
\end{aligned} \tag{62}$$

where $\dot{\mathbf{R}}_u(\hat{t}) \in \mathbb{R}^{2\times2}$ and $\dot{\boldsymbol{w}}_u(\hat{t}) \in \mathbb{R}^2$ are the gradients w.r.t $\hat{t}$. In practice, trajectories of other obstacles are fitted by piecewise polynomials. Therefore, the position $\boldsymbol{w}_u(\hat{t})$, the rotation matrix $\mathbf{R}_u(\hat{t})$ and their gradients can also be calculated analytically. Besides, we set α as 100 to approximate the maximum function and -100 to smooth the minimum function.

VI. Reformulation of Trajectory Optimization

In this section, we analyze the characteristics of the constraints Eq.(5c-5f)(9) in trajectory planning and use targeted approaches to eliminate them, respectively. Then, the original optimization problem is reformulated into an unconstrained program that can be further solved efficiently.

A. Feasibility Constraints

We adopt the discrete-time summation-type penalty term S_Σ to relax the feasibility constraints Eq. (9):

$$\begin{aligned} S_\Sigma(\mathbf{c},\mathbf{T}) &= \sum_{d\in\mathcal{D}} w_d \sum_{i=1}^{n}\sum_{j=1}^{M_i}\sum_{k=0}^{\lambda} P_{d,i,j,k}(\boldsymbol{c}_{i,j},\mathbf{T}), \\ P_{d,i,j,k}(\boldsymbol{c}_{i,j},\mathbf{T}) &= \frac{\delta T_i}{\lambda}\bar{\omega}_k \mathrm{L}_1(\mathcal{G}_{d,i,j,k}), \end{aligned} \tag{63}$$

where w_d is the penalty weight corresponding to different kinds of constraints. $[\bar{\omega}_0,\bar{\omega}_1,...,\bar{\omega}_{\lambda-1},\bar{\omega}_\lambda] = [1/2,1,...,1,1/2]$ are the quadrature coefficients from the trapezoidal rule [50] and $P_{d,i,j,k}$ is the violation penalty for a constraint point. Moreover, we define a first-order relaxation function $\mathrm{L}_1(\cdot)$ to guarantee the continuous differentiability and non-negativity of penalty terms:

$$\mathrm{L}_1(x) = \begin{cases} 0 & x \le 0, \\ -\dfrac{1}{2a_0^3}x^4 + \dfrac{1}{a_0^2}x^3 & 0 < x \le a_0 \\ x - \dfrac{a_0}{2} & a_0 < x. \end{cases} \tag{64}$$

Here $a_0 = 10^{-4}$ is the demarcation point. Such discrete penalty formulation ensures that continuous-time constraints Eq.(5g) are satisfied within an acceptable tolerance. Then, the trajectory optimization for vehicles is reformatted as follows:

$$\min_{\mathbf{c},\mathbf{T}} \mathcal{J}(\mathbf{c},\mathbf{T}) = J(\mathbf{c},\mathbf{T}) + S_\Sigma(\mathbf{c},\mathbf{T}) \tag{65a}$$

$$\text{s.t.}\boldsymbol{\sigma}_0^{[s-1]}(0) = \bar{\boldsymbol{\sigma}}_0,\ \boldsymbol{\sigma}_n^{[s-1]}(T_n) = \bar{\boldsymbol{\sigma}}_f, \tag{65b}$$

$$\boldsymbol{\sigma}_i^{[s-1]}(T_i) = \boldsymbol{\sigma}_{i+1}^{[s-1]}(0) = \widetilde{\boldsymbol{\sigma}}_i, 1 \le i < n, \tag{65c}$$

$$\boldsymbol{\sigma}_{i,j}^{[\widetilde{d}]}(\delta T_i) = \boldsymbol{\sigma}_{i,j+1}^{[\widetilde{d}]}(0), 1 \le i \le n, 1 \le j < M_i, \tag{65d}$$

$$T_i > 0, 1 \le i \le n. \tag{65e}$$

Without loss of generality, the gradients of the violation penalty at each constraint point on the trajectory are derived:

$$\begin{aligned} \frac{\partial P_{d,i,j,k}}{\partial \boldsymbol{c}_{i,j}} &= \frac{\partial P_{d,i,j,k}}{\partial \mathcal{G}_{d,i,j,k}}\frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \boldsymbol{c}_{i,j}}, \\ \frac{\partial P_{d,i,j,k}}{\partial \mathbf{T}} &= \left(\mathbf{0}_{i-1}^{\mathrm{T}}, \frac{P_{d,i,j,k}}{T_i}, \mathbf{0}_{n-i}^{\mathrm{T}}\right)^{\mathrm{T}} + \frac{\partial P_{d,i,j,k}}{\partial \mathcal{G}_{d,i,j,k}}\frac{\partial \mathcal{G}_{d,i,j,k}}{\partial \mathbf{T}}, \\ \frac{\partial P_{d,i,j,k}}{\partial \mathcal{G}_{d,i,j,k}} &= \frac{\delta T_i}{\lambda}\bar{\omega}_k \mathrm{L}_1'(\mathcal{G}_{d,i,j,k}). \end{aligned} \tag{66}$$

Since the gradients of the constraints $\mathcal{G}_{d,i,j,k}$ have been calculated in Sect. IV and Sect. V, we can also analytically obtain the gradients of the violation penalty $\partial P_{d,i,j,k}$ by the above propagation chain Eq.(66). Then, the gradients of the newly defined objective function $\mathcal{J}(\mathbf{c},\mathbf{T})$ can be easily obtained in a similar way.

B. Continuity Constraints

For piece-wise polynomials, high-order continuity is required at the connection points (waypoints $\boldsymbol{q}_i$) of each piece to ensure smoothness of motion, as shown in Eq.(65d). Based on the optimality condition [51], in our problem, the minumum control effort piece-wise polynomial should be snap-continuous at the waypoints. Accordingly, when the start and end states of the piece-wise polynomial are fixed, the parameter space can be transformed from coefficients and duration to waypoints and duration. Then, the continuity constraint Eq.(65d) can be naturally satisfied.

C. Gear Shifting Constraints

In the previous section, we eliminate continuity constraints without sacrificing the optimality condition by transforming the polynomial parameter space. However, the start and end states of piece-wise polynomials are required to be fixed, which means the states at GSP are fixed. However, if the states at GSP are entirely determined by the initial guess and not involved in optimization, this undoubtedly restricts the trajectory's freedom and strengthens the dependence on the initial guess, leading to lower trajectory quality. Therefore, in this section, we improve the previous work [51] by introducing GSPO to further relax the states at GSP, enabling the planner to better handle complex driving scenarios that require multiple forward and backward movements.

Before discussing GSPO and the elimination of the gear shifting constraints, we first decompose and reformulate Eq.(65c):

$$\boldsymbol{\sigma}_i(T_i) = \boldsymbol{\sigma}_{i+1}(0) = \boldsymbol{p}_i^g, \tag{67a}$$

$$\boldsymbol{\sigma}_i^{(1)}(T_i) = -\boldsymbol{\sigma}_{i+1}^{(1)}(0) = \boldsymbol{v}_i^g, \tag{67b}$$

$$\boldsymbol{\sigma}_i^{(2)}(T_i) = \boldsymbol{\sigma}_{i+1}^{(2)}(0) = \mathbf{0}_2, \tag{67c}$$

$$||\boldsymbol{v}_i^g||_2 = \bar{v}, \tag{67d}$$

$$\forall i \in \{1,2,3,...,n-1\}, \forall \bar{d} \in \{2,...,s-1\}. \tag{67e}$$

p_i^g is the gear shifting position and v_i^g is the final velocity before the shift. It is worth noting that the velocity direction is reversed before and after the shift and its magnitude is set to a small non-zero value $\bar{v}$ to prevent singularities during optimization.To ensure comfort, the acceleration at GSP is fixed to zero and does not participate in the subsequent relaxation process. The key to introducing GSPO is to add the states at GSP as independent free optimization variables to the trajectory representation space. To facilitate subsequent gradient-based solving, we need to derive the relationship and gradient propagation chain between p_i^g, v_i^g and polynomial parameters (coefficients and duration).

Based on the Eq.(67a-67c) and the continuity guarantee of piece-wise polynomials mentioned in Sect.VI-B, our unique trajectory representation essentially satisfies the following linear equation:

$$\mathbf{M}_i(T_i)\boldsymbol{c}_i = \mathbf{b}_i, 1 \le i \le n, \tag{68}$$

where $\mathbf{M}_i(T_i) \in \mathbb{R}^{2M_i s \times 2M_i s}$ is an invertible banded matrix whose specific form can be found in [51]. $\mathbf{b}_{i\in\{1...n\}} \in \mathbb{R}^{2M_i s \times 2}$ is defined as follows:

$$\begin{aligned} \mathbf{b}_1 &= \left(\bar{\boldsymbol{\sigma}}_0, \boldsymbol{q}_{1,1}, \mathbf{0}_{2\times\widetilde{d}}, ..., \boldsymbol{q}_{1,M_1-1}, \mathbf{0}_{2\times\widetilde{d}}, p_1^g, v_1^g, \mathbf{0}_{2\times(s-2)}\right)^{\mathrm{T}}, \\ \mathbf{b}_n &= \left(p_{n-1}^g, v_{n-1}^g, \mathbf{0}_{2\times(s-2)}, \boldsymbol{q}_{n,1}, \mathbf{0}_{2\times\widetilde{d}}, ..., \boldsymbol{q}_{n,M_n-1}, \mathbf{0}_{2\times\widetilde{d}}, \bar{\boldsymbol{\sigma}}_f\right)^{\mathrm{T}}. \\ \mathbf{b}_i &= \left(p_{i-1}^g, v_{i-1}^g, \mathbf{0}_{2\times(s-2)}, \boldsymbol{q}_{i,1}, \mathbf{0}_{2\times\widetilde{d}}, ...,\right. \\ &\qquad \left.\boldsymbol{q}_{i,M_i-1}, \mathbf{0}_{2\times\widetilde{d}}, p_i^g, v_i^g, \mathbf{0}_{2\times(s-2)}\right)^{\mathrm{T}}, 1 < i < n, \end{aligned} \tag{69}$$

where the degree of continuity $\widetilde{d}$ is set to $2s-1$ to satisify the optimality condition. Since the physical meaning of GSP

is the connection point between the preceding and succeeding piece-wise polynomials, it is intuitive that the states at GSP are related to the parameters of both the preceding and succeeding piece-wise polynomials. Therefore, the gradient of the objective function w.r.t the states at GSP should be propagated through the gradients of both adjacent piece-wise polynomials. Here, we derive the gradients w.r.t the gear shifting position $\boldsymbol{p_i^g}$ and the velocity $\boldsymbol{v_i^g}$:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{p_i^g}} = \left(\mathbf{M}_i^{-\mathrm{T}} \frac{\partial \mathcal{J}}{\partial \boldsymbol{c}_i}\right)^{\mathrm{T}} e_{(2M_i-1)s+1} + \left(\mathbf{M}_{i+1}^{-\mathrm{T}} \frac{\partial \mathcal{J}}{\partial \boldsymbol{c}_{i+1}}\right)^{\mathrm{T}} e_1, \quad (70)$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{v_i^g}} = \left(\mathbf{M}_i^{-\mathrm{T}} \frac{\partial \mathcal{J}}{\partial \boldsymbol{c}_i}\right)^{\mathrm{T}} e_{(2M_i-1)s+2} + \left(\mathbf{M}_{i+1}^{-\mathrm{T}} \frac{\partial \mathcal{J}}{\partial \boldsymbol{c}_{i+1}}\right)^{\mathrm{T}} e_2, \quad (71)$$

$$\forall i \in \{1, 2, 3, ..., n-1\}.$$

e_k is a column vector of proper dimension, where the element of the k-th row is 1 and all others are 0. So far, we successfully add $\boldsymbol{p_i^g}$ and $\boldsymbol{v_i^g}$ into the trajectory representation space. Then, our trajectory representation space evolves from the coefficients and duration to waypoints, duration, $\boldsymbol{p_i^g}$ and $\boldsymbol{v_i^g}$ and naturally satisfies these equality constraints Eq.(67a-67c). Moreover, the original trajectory optimization problem is reformulated as follows:

$$\min_{\mathbf{q},\mathbf{T},\mathbf{p}^g,\mathbf{v}^g} \mathcal{J}\left(\mathbf{q}, \mathbf{T}, \mathbf{p}^g, \mathbf{v}^g\right) = \mathcal{J}\left(\mathbf{c}\left(\mathbf{q}, \mathbf{T}, \mathbf{p}^g, \mathbf{v}^g\right), \mathbf{T}\right) \quad (72a)$$

$$\text{s.t.} ||\boldsymbol{v}_i^g||_2 = \bar{v}, 1 \le i \le n, \quad (72b)$$

$$T_i > 0, 1 \le i \le n. \quad (72c)$$

Here, waypoints $\mathbf{q} = (\boldsymbol{q}_1, ..., \boldsymbol{q}_n) \in \mathbb{R}^{2\times\sum_{i=1}^{n}(M_i-1)}$, gear shifting positions $\mathbf{p}^g = \left(\boldsymbol{p}_1^g, ..., \boldsymbol{p}_{n-1}^g\right) \in \mathbb{R}^{2\times(n-1)}$ and gear shifting velocity $\mathbf{v}^g = \left(\boldsymbol{v}_1^g, ..., \boldsymbol{v}_{n-1}^g\right) \in \mathbb{R}^{2\times(n-1)}$. Furthermore, to satisfy the constraint Eq.(72b), we express $\boldsymbol{v}_i^g = \left(\bar{v}\cos\theta_i^g, \bar{v}\sin\theta_i^g\right)^{\mathrm{T}}$ in polar coordinates, where the physical meaning of θ_i^g is the direction of velocity before gear switching. Besides, we define the angle set $\boldsymbol{\theta}^g = \left(\theta_1^g, \theta_2^g, ..., \theta_{n-1}^g\right)^{\mathrm{T}} \in \mathbb{R}^{n-1}$ for subsequent derivation and the cost function is transformed to $\mathcal{K}\left(\mathbf{q}, \mathbf{T}, \mathbf{p}^g, \boldsymbol{\theta}^g\right) = \mathcal{J}\left(\mathbf{q}, \mathbf{T}, \mathbf{p}^g, \mathbf{v}^g\left(\boldsymbol{\theta}^g\right)\right)$. The gradient of the cost function $\mathcal{K}$ w.r.t $\boldsymbol{\theta}^g$ can be obtained as follows:

$$\frac{\partial \mathcal{K}}{\partial \theta_i^g} = \left(-\bar{v}\sin\theta_i^g, \bar{v}\cos\theta_i^g\right) \frac{\partial \mathcal{J}}{\partial \boldsymbol{v}_i^g}, 1 \le i < n. \quad (73)$$

At this point, the last equality constraint Eq.(72b) is also removed. In summary, the key idea of GSPO is to transform the parameter space of the trajectory and ultimately represent the vehicle trajectory with complex forward-backward motion using waypoints, duration, $\boldsymbol{p}_i^g$, and θ_i^g. This method effectively achieves GSPO while eliminating all the equality constraints present in the original problem. Furthermore, we evaluate the effect of GSPO on trajectory quality and efficiency, as shown in the ablation experiments in Sect.VII-C.

D. Positiveness Condition

To remove the strict positiveness condition Eq.(72c), we define an unconstrained virtual time $\boldsymbol{\tau} = [\tau_1, ..., \tau_i, ..., \tau_n]^{\mathrm{T}} \in \mathbb{R}^n$ and employ a diffeomorphism map [52] from $\tau_i \in \mathbb{R}$ to the real duration $T_i \in \mathbb{R}^+$:

$$T_i = \begin{cases} \frac{1}{2}\tau_i^2 + \tau_i + 1 & \tau_i > 0 \\ \frac{2}{\tau_i^2 - 2\tau_i + 2} & \tau_i \le 0 \end{cases} \quad (74)$$

$$\forall i \in \{1, 2, 3, ..., n\}.$$

Then, we use $\boldsymbol{\tau}$ to replace the real time set $\mathbf{T}$ so that the strict positiveness constraints Eq.(72c) are naturally satisfied. The corresponding reformulated optimization problem is as follows:

$$\min_{\mathbf{q},\boldsymbol{\tau},\mathbf{p}^g,\boldsymbol{\theta}^g} \mathcal{W}\left(\mathbf{q}, \boldsymbol{\tau}, \mathbf{p}^g, \boldsymbol{\theta}^g\right) = \mathcal{K}\left(\mathbf{q}, \mathbf{T}\left(\boldsymbol{\tau}\right), \mathbf{p}^g, \boldsymbol{\theta}^g\right). \quad (75)$$

The gradient can be propagated from T_i to τ_i:

$$\frac{\partial \mathcal{W}}{\partial \tau_i} = \begin{cases} (\tau_i + 1)\frac{\partial \mathcal{K}}{\partial T_i} & \tau_i > 0 \\ \frac{4(1-\tau_i)}{\left(\tau_i^2 - 2\tau_i + 2\right)^2}\frac{\partial \mathcal{K}}{\partial T_i} & \tau_i \le 0 \end{cases} \quad (76)$$

$$\forall i \in \{1, 2, 3, ..., n\}.$$

The gradients w.r.t $\boldsymbol{\tau}$ are efficiently calculated once we obtain the gradients w.r.t $\mathbf{T}$.

In summary, we have removed all constraints in the optimization and analytically derived the gradients. Finally, we robustly solve the transformed unconstrained optimization Eq.(75) via the quasi-Newton method [53].

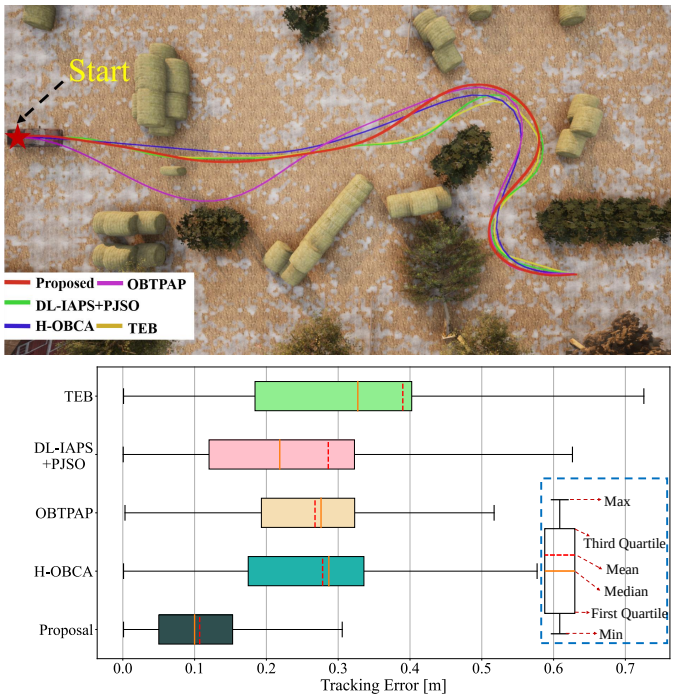


Fig. 5: The visualization of trajectories and their corresponding tracking errors. The top figure depicts the intuitive visualization of various planned trajectories. The bottom figure shows the statistical analysis of tracking errors for each trajectory during their execution in this case.

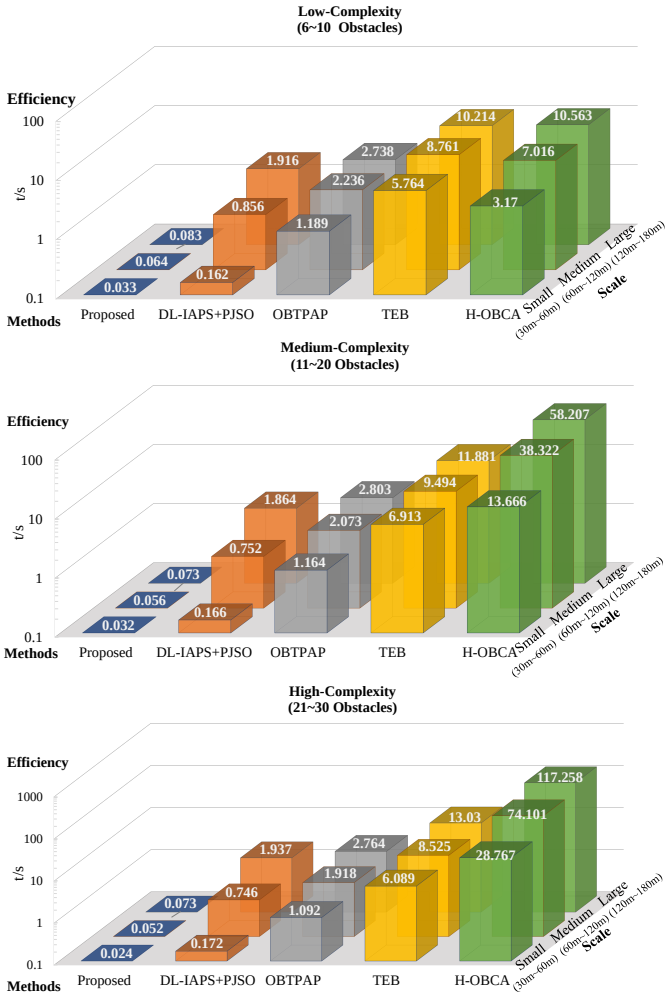


Fig. 6: The visualization of time efficiency comparison across various cases, which demonstrates the significant advantage of our proposed method.

VII. Evaluations

The proposed method is evaluated both in simulation and in the real world. Benchmarks show that our planner outperforms state-of-the-art methods in terms of time efficiency and trajectory quality. In the detailed implementation, we avoid singularities by fixing the speed magnitude to a non-zero small value ($\bar{v} = 0.05m/s$) when switching between forward and reverse gears. Although this treatment leads to a sudden change in velocity, the change is so slight that the negative impact on control is negligible. The subsequent tracking error comparison experiments and physical experiments also verify the dynamical executability of our planned trajectory.

A. Benchmarks

1) Configurations: All simulation experiments are conducted on a desktop computer running Ubuntu 18.04 with an Intel Core i7-10700 CPU and a GeForce RTX 2060 GPU.

We perform simulation experiments based on the physical simulator CARLA [54]. The proposed method is compared with four impressive methods specifically designed for motion planning of car-like robots in static unstructured environments, including OBTPAP [33], DL-IAPS+PJSO [28], H-OBCA [31] and Timed Elastic Bands (TEB) [55]. All methods use the bicycle model and are implemented in C++14 without parallel acceleration. OBTPAP [33] and H-OBCA [31] are solved by primal-dual interior-point method IPOPT [56]. DL-IAPS+PJSO [28] is implemented using OSQP [57]. The graph optimization solver G^2o [58] is used for TEB [55]. Moreover, for fair comparisons, all planners use hybridA* [29] algorithm as their front-end to provide rough initial guesses for the subsequent optimization. Additionally, the proposed planner, OBTPAP [33], DL-IAPS+PJSO [28] and TEB [55] use known global point clouds in the environment to construct collision-free constraints. H-OBCA [31] uses known convex polygons manually extracted from obstacles to construct its optimization formulation. All common parameters, including convergence conditions, dynamics constraints, and the ego vehicle dimensions, are set to the same for fairness.

2) Quantitative Results: We conduct extensively quantitative assessments in many cases with different numbers of environmental obstacles and start-end distances (denote problem scale). Lots of comparison tests are performed in each case with random starting and ending states. To evaluate the execution performance of trajectories, we employ an MPC controller [59] that minimizes position and velocity errors to track planned trajectories and measure the tracking error. To provide a more intuitive understanding, we plot the planned trajectories and corresponding tracking error in one experiment, as shown in Fig. 5. During each experiment, we record the physical dynamics data of the planned trajectories, including mean acceleration (M.A.), jerk (M.JK), and tracking error (M.TE), as presented in Tab. I. We also calculate the average planning time for each case and plot it on a logarithmic histogram, as shown in Fig. 6. Based on the simulation results, we believe that our method has significant advantages over other methods in terms of trajectory execution performance, motion comfort, and time efficiency. 1) In terms of tracking error, both Fig. 5 and Tab. I demonstrate that our planned trajectories exhibit the lowest tracking error, indicating their superior execution performance when executed by the lower-level controller. We attribute this to the powerful spatial-temporal optimal deformation capability of our trajectories and the effective high-order continuous trajectory representation. Compared to methods that employ discrete motion processes, our approach essentially guarantees the continuity of the state and finite-dimensional derivatives, making our trajectory closer to the real physical motion process and thus easier to execute. 2) In terms of dynamic metrics, Tab. I shows that our trajectories exhibit low acceleration and jerk, indicating better human comfort [44]. 3) In terms of time efficiency, the histogram in Fig. 1 reveals that our algorithm has an order-of-magnitude efficiency advantage over other algorithms in each case, which is particularly pronounced in large-scale trajectory generation problems. Furthermore, the results demonstrate the robustness of our method against problem scale and the number of environmental obstacles, enabling it to adapt to different scenarios. It is worth mentioning that because H-OBCA [31] directly introduces dual variables to accurately

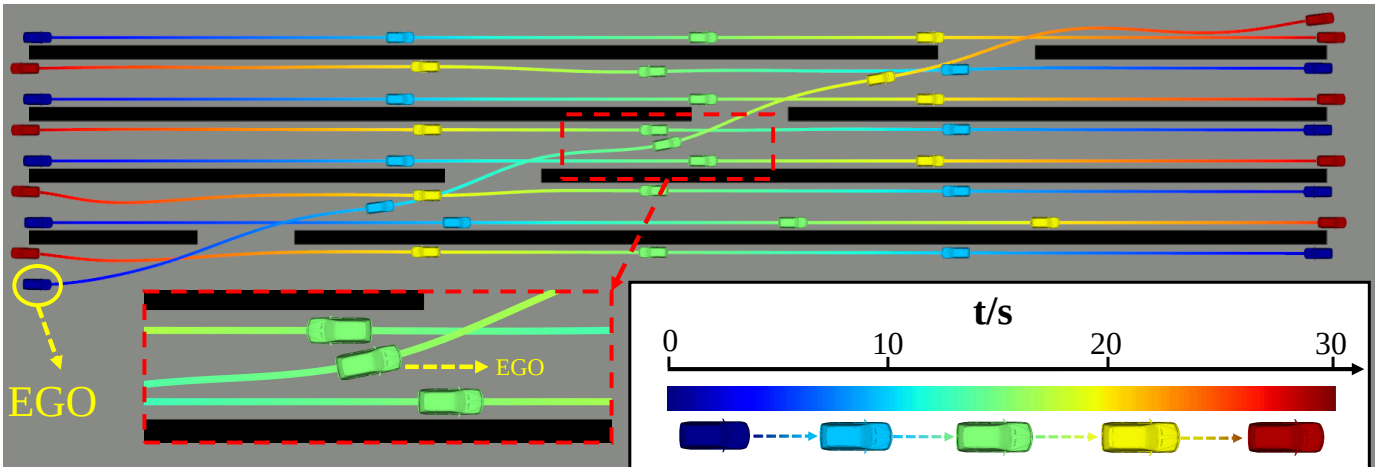


Fig. 7: Motion visualization in the dynamic environment, where the colors represent timestamps.

encode the distance to obstacles, it can also generate high-quality trajectories. However, in terms of time efficiency and robustness, our method outperforms H-OBCA [31], as its computation burden makes it intolerable to be applied in real-time scenarios, especially in dense environments.

B. Dynamic Obstacle Avoidance

We also validate the proposed planner in a $210m \times 50m$ dynamic unstructured environment where the ego vehicle is required to avoid static obstacles and the traffic flow consisting of eight other vehicles, as shown in Fig. 7. Since perception is not the focus of this paper, the environment and the trajectories of other vehicles are known to the planner. The maximum velocity of all vehicles is set to $10m/s$. The colors of vehicles and trajectories in Fig. 7 represent motion timestamps, which indicate the absence of spatial-temporal intersections between the ego vehicle and other moving objects, demonstrating dynamic collision-free property. Thanks to full-dimensional modeling of objects, the ego vehicle has the ability to fully utilize the safe space to get through narrow areas, as the close-up shows. The trajectory length of the ego vehicle is $260m$, while the computation time is only $0.18s$, which demonstrates the efficiency of our planner in complex dynamic environments, especially for long-distance global trajectory generation.

C. Ablation Experiments

To evaluate the effectiveness of our proposed gear shifting point optimization (GSPO) approach for trajectory planning, we conduct ablation experiments to compare the results with and without GSPO. We perform numerous trajectory planning experiments with diverse endpoints in a cluttered environment, where each trajectory involves multiple forward and backward movements, as shown in Fig. 8. Since the formulation of the objective function ,$J = \int_0^{T_s} \ddot{\boldsymbol{\sigma}}(t)^{\mathrm{T}} \ddot{\boldsymbol{\sigma}}(t) dt + w_T T_s$, remains the same with or without GSPO, we use it to represent the trajectory quality in this experiment. Here, w_T is set to 10. Moreover, we measure the computation time of the planner for each planning instance to evaluate the impact of GSPO on

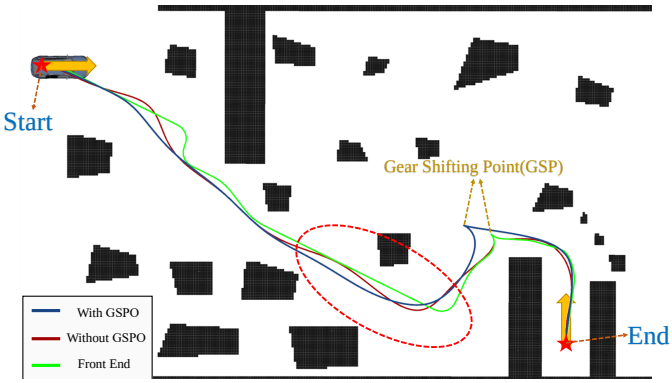


Fig. 8: Trajectory visualization with or without GSPO. Without GSPO, the degree of freedom in trajectory planning is limited, potentially resulting in the generation of the undesirable back-and-forth curved trajectory, as illustrated by the dark red line within the red ellipse. In this case, the loss value of the trajectory with GSPO is 385, while that without GSPO is 591. This quantitatively demonstrates the improvement in trajectory quality resulting from GSPO.

the computational efficiency. To better reflect the comparison, we define the time reduction rate as $\frac{t^c_{withoutGSPO} - t^c_{withGSPO}}{t^c_{withoutGSPO}}$ and the quality improvement rate as $\frac{J_{withoutGSPO} - J_{withGSPO}}{J_{withoutGSPO}}$. The quantitative comparison result of the ablation experiment is presented in Tab. II. The result indicates that GSPO can significantly improve trajectory quality and reduce computation time. Specifically, the introduction of GSPO enhances the degree of freedom in trajectory optimization and reduces dependence on initial values, leading to a more effective trajectory optimization process. Moreover, GSPO enables a more comprehensive exploration of the solution space by the planner, resulting in further enhancements in trajectory quality.

D. Real-World Experiments

In addition to simulations, we also conduct real-world experiments to verify the feasibility of our planner on a real

TABLE I: The comparative analysis of dynamical statistics across various cases, which effectively demonstrates the advantages of our proposed method in terms of human comfort and execution performance.

Environments	Problem Scale	Small-Scale ($30m \sim 60m$)			Medium-Scale ($60m \sim 120m$)			Large-Scale ($120m \sim 180m$)		
	Method	M.A (m/s^2)	M.JK (m/s^3)	M.TE (m)	M.A (m/s^2)	M.JK (m/s^3)	M.TE (m)	M.A (m/s^2)	M.JK (m/s^3)	M.TE (m)
Low-Complexity ($6 \sim 10$ Obstacles)	Proposed	**6.96**	**16.61**	**0.142**	**6.76**	**18.81**	**0.131**	**6.03**	**15.30**	**0.146**
	OBTPAP	30.11	101.99	0.223	51.32	157.19	0.231	36.10	105.56	0.250
	DL-IAPS+PJSO	10.02	66.44	0.208	9.40	77.79	0.240	10.14	86.09	0.252
	H-OBCA	7.42	20.67	0.194	8.23	24.11	0.217	6.40	19.62	0.258
	TEB	16.27	380.52	0.348	15.74	427.35	0.338	14.96	363.07	0.321
Medium-Complexity ($11 \sim 20$ Obstacles)	Proposed	**7.50**	**20.69**	**0.142**	**7.60**	**18.66**	**0.135**	7.17	**17.68**	**0.115**
	OBTPAP	42.27	145.89	0.239	58.01	197.21	0.236	49.94	158.09	0.251
	DL-IAPS+PJSO	11.53	83.92	0.223	13.01	119.75	0.257	14.26	140.55	0.254
	H-OBCA	7.85	21.69	0.216	8.88	26.25	0.213	**7.05**	20.57	0.231
	TEB	18.72	466.93	0.348	21.94	614.42	0.343	21.32	707.91	0.357
High-Complexity ($21 \sim 30$ Obstacles)	Proposed	**7.53**	**17.95**	**0.127**	**7.97**	**19.49**	**0.120**	**6.97**	**17.56**	**0.126**
	OBTPAP	36.54	128.29	0.230	43.90	157.32	0.247	57.03	193.75	0.251
	DL-IAPS+PJSO	12.25	94.43	0.210	16.24	155.38	0.259	13.92	134.74	0.268
	H-OBCA	8.09	23.58	0.195	8.17	25.77	0.230	7.05	21.49	0.248
	TEB	19.67	493.70	0.352	23.42	607.32	0.416	19.89	584.01	0.365

TABLE II: Ablation Experiments With and Without GSPO

Procedure	Without GSPO	With GSPO
Computation Time	0.054 s	0.048 s
Reduction Rate	-	**11.1**%
Loss Value	503.0	398.9
Improvement Rate	-	**20.7**%

Fig. 9: The real-world experiment site.

physical platform. The experimental site is a dense $40m \times 20m$ outdoor unstructured parking lot, as shown in Fig. 9. The moving robot is required to start from an initial state, go around obstacles, and eventually reverse into a parking space at a human-defined heading angle. The length of the whole track is about 42 meters. Real-world experiments are conducted on a SAIC-GM-Wuling Automobile Baojun E300[2] with dimensions of $2.9m \times 1.7m \times 1.6m$, a wheelbase of $2m$ and no GPS included, as shown in Fig. 10. A set of sensory elements

Fig. 10: The experimental platform.

including one stereo camera, four fisheye cameras, and twelve ultrasonic sensors are deployed on the platform for real-time localization, control, and data recording. Additionally, neither high precision LIDAR nor precise external positioning systems are used in our real-world experiments. All modules are implemented in C++ and deployed on TI TDA4 Chip (64-bit Dual Arm Cortex-A72 @2000MHz) [3] configured with QNX Operating System. During the trajectory planning, the vehicle shape was expanded by $0.25m$ to ensure collision-free property during actual execution in the presence of unavoidable control and positioning errors. The maximum forward and backward speeds are set to $2m/s$ and $0.5m/s$, respectively. The time weight w_T is set to 50 to ensure the aggressiveness of the trajectory. The motion of the ego vehicle is visualized in Fig. 11, which demonstrates that the ego vehicle can follow the planned trajectory to avoid obstacles and eventually reverse into a parking space with the user-given heading angle. Furthermore, dynamic evaluation metrics

[2]https://www.sgmw.com.cn/E300.html/

[3]https://www.ti.com/product/TDA4VM

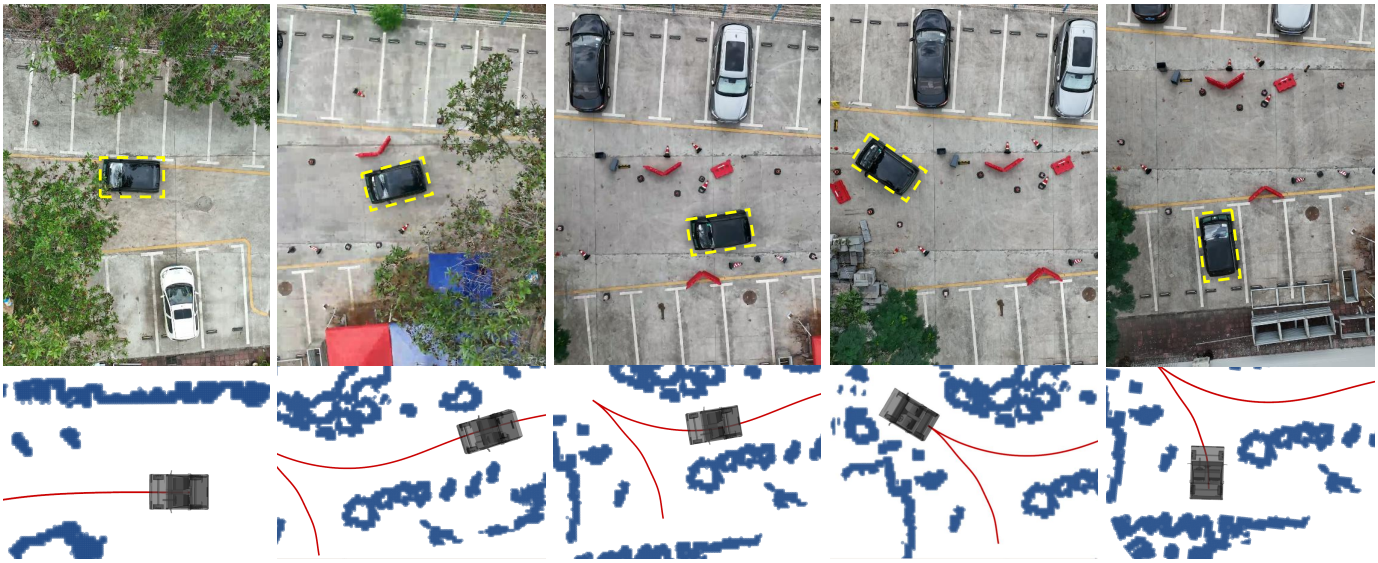


Fig. 11: The motion diagram of the ego vehicle in the real-world experiment, where the red curve is the execution trajectory.

TABLE III: Dynamic Statistics in Real-World Experiments

Statistics	Mean	Max	STD.
Forward.Vel. (m/s)	1.53	1.95	0.52
Backward.Vel. (m/s)	0.45	0.50	0.10
Forward.Jerk. (m/s^3)	0.25	6.95	0.50
Backward.Jerk (m/s^3)	0.18	7.07	0.55

are quantified in Tab. III. As we can see, the ego vehicle maintains a relatively high speed throughout to reach the target state without exceeding the dynamic limits while keeping a low jerk to ensure the comfort of passengers. Finally, more demonstrations can be found in the supplementary material.

VIII. Conclusion

In this paper, we propose an efficient spatial temporal optimal planning system based on the differential flatness property. Our method possesses several key features: Firstly, our trajectory representation is inherently high-order continuous and does not discretize the motion process, allowing it to better approximate the true motion process. Secondly, unlike point mass models, we implement convex approximations for obstacle avoidance to ensure sufficient utilization of the solution space. Thirdly, we relax the original optimization problem and introduce state optimization targeted at gear-switching points to increase the robustness towards initial values. Our method achieves a good balance between efficiency and trajectory quality, with simulation experiments demonstrating that our method has the highest efficiency while maintaining the lowest tracking error. This means that our generated trajectories are sufficiently smooth to be executed well by low-level controllers. Real-world experiments also validate the effectiveness of our method on a real platform.

In the future, we plan to extend the application of our method to multiple robots and explore its applicability to other motion models. Furthermore, we will direct our efforts towards motion planning for challenging terrains, such as uphill and downhill trajectories. Our ultimate goal is to develop a comprehensive motion planning framework that can be applied to a wide range of robotic platforms and environments.

Attachments

TABLE IV: Statistics of Trajectory Dynamics under Different Uncertainties

Uncertainty	Free	Low	Medium	High
Delay(s)	0	0.02	0.05	0.07
Noise	0	[-0.05,0.15]	[-0.075, 0.225]	[-0.1,0.3]
R.N	3	9	17	56
M.TE (m)	0.044	0.072	0.087	0.098
T.L (m)	64.3	65.2	65.5	65.9
M.A (m/s^2)	0.81	0.87	1.06	1.28
M.SR (s^{-1})	0.32	0.37	0.48	0.63

We conduct simulation experiments in a $50m * 10m$ unknown environment to evaluate the impact of uncertainty on autonomous navigation, as shown in Fig. 12. The robot, with dimensions of $0.8m * 0.5m$, is equipped with an omnidirectional radar capable of sensing up to $10m$. After the planner outputs the trajectory, we use an MPC-based controller to follow it. The kinematic model with noise n used in this experiment is as follows:

$$\dot{\mathrm{x}} = f(\mathrm{x}, \mathrm{u}, n), \tag{77a}$$

$$\mathrm{x} = [p_x, p_y, v, \phi, \theta]^{\mathrm{T}}, \tag{77b}$$

$$\mathrm{u} = [a, \omega]^{\mathrm{T}}, \tag{77c}$$

where $[p_x, p_y]$ is the position of the rear center, v is the longitude velocity, ϕ is the steer angle, θ is the yaw angle,

a is the longitude acceleration, ω is the steer angle rate and L_w is the wheelbase. Here, the state transfer equation f with noise is defined as:

$$f(\mathrm{x}, \mathrm{u}, n) = q(\mathrm{x}, \mathrm{u}) + N(\mathrm{x}, \mathrm{u}, n), \tag{78a}$$

$$q(\mathrm{x}, \mathrm{u}) = [v\cos\theta, v\sin\theta, a, \omega, v\tan\phi/L_w]^{\mathrm{T}}, \tag{78b}$$

q is the theoretical state transfer and $N = nq$ is the uncertainty term defined as proportional to the change in motion. n is a diagonal matrix representing the intensity of random noise, where each element satisfies a uniform distribution on $[h_1, h_2]$. Replanning is triggered if either of the following two conditions is met: 1) the original trajectory collides with a newly detected obstacle, or 2) the tracking error exceeds $0.1m$. The number of replans triggered by the second condition is denoted as R.N. To better simulate real-world scenarios, delays are introduced between the controller and actuator in this experiment. The experiment cases are classified into four groups based on the levels of noise and delay: free uncertainty, low uncertainty, medium uncertainty, and high uncertainty. For each case, the R.N, mean tracking error (M.TE), trajectory length (T.L), mean acceleration (M.A), and mean steering angle rate (M.SR) are calculated. The quantitative results are summarized in Tab. IV. These results indicate that increased uncertainty leads to three negative effects: 1) elevated tracking error, indicating greater difficulty in executing the trajectory; 2) increased mean acceleration and steering rate, suggesting higher control effort; and 3) longer trajectories, signifying more curved motion and reduced optimality. Despite the presence of uncertainty, the robot exhibits resilience due to its high-frequency controller feedback and real-time replanning capability, enabling the safe navigate in unknown environments. Additionally, the results presented in Tab. IV indicate that an increase in uncertainty leads to a corresponding rise in the number of replans caused by excessive tracking error. The fast replanning mechanism allows the robot to correct its trajectory deviations and eliminate accumulated tracking errors, enhancing its robustness against uncertainty. Certainly, we acknowledge that incorporating the potential uncertainty into the controller can enhance the system's overall resilience to uncertainty. Additionally, considering uncertainty during the planning phase can also make the trajectory easier to follow, which will be addressed in our future work.

REFERENCES

[1] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, 2019.

[2] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 341–355, 2019.

[3] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous driving on curvy roads without reliance on frenet frame: A cartesian-based trajectory planning method," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[4] A. Gupta and R. Divekar, "Autonomous parallel parking methodology for ackerman configured vehicles," *ACEEE International Journal on Communication*, vol. 1, no. 2, pp. 1–6, 2010.

[5] C. Sungwoo, C. Boussard, and B. d'Andréa Novel, "Easy path planning and robust control for automatic parallel parking," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 656–661, 2011.

[6] K. Kondak and G. Hommel, "Computation of time optimal movements for autonomous parking of non-holonomic mobile platforms," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, vol. 3. IEEE, 2001, pp. 2698–2703.

[7] H. Shin, D. Kim, and S.-E. Yoon, "Kinodynamic comfort trajectory planning for car-like robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6532–6539.

[8] B. Li and Z. Shao, "Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots," *Advances in Engineering Software*, vol. 87, pp. 30–42, 2015.

[9] 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, 2015.

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

[11] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[12] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1879–1884.

[13] M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3071–3077.

[14] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4889–4895.

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

[16] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2859–2865.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[18] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5054–5061.

[19] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on RRT," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5622–5627.

[20] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2775–2781.

[21] N. Seegmiller, J. Gassaway, E. Johnson, and J. Towler, "The maverick planner: An efficient hierarchical planner for autonomous vehicles in unstructured environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2018–2023.

[22] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 5628–5635.

[23] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8154–8161, 2022.

[24] Z. Alabedeen Ali and K. Yakovlev, "Safe interval path planning with kinodynamic constraints," *arXiv e-prints*, pp. arXiv–2302, 2023.

[25] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[26] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *2015 54th IEEE conference on decision and control (CDC)*. IEEE, 2015, pp. 835–842.

[27] 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, 2018.

[28] J. Zhou, R. He, Y. Wang, S. Jiang, Z. Zhu, J. Hu, J. Miao, and Q. Luo, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 439–446, 2021.

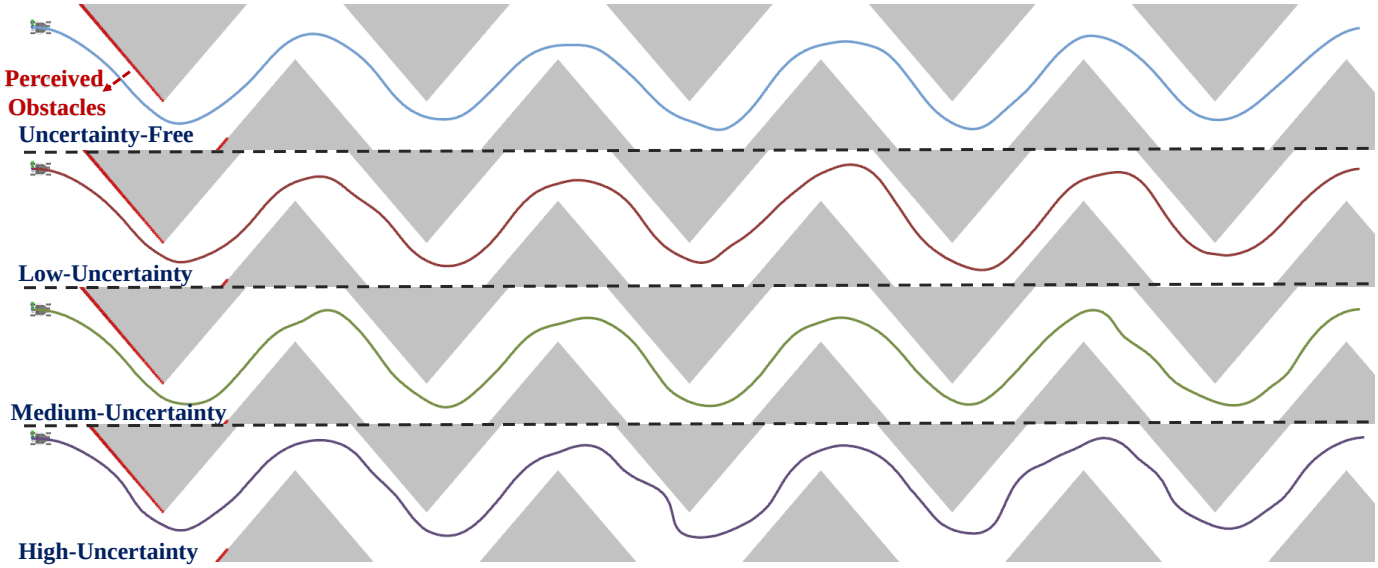


Fig. 12: Motion trajectory visualization under each case, where the gray areas represent obstacles.

[29] 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, 2010.

[30] S. Shi, Y. Xiong, J. Chen, and C. Xiong, "A bilevel optimal motion planning (bomp) model with application to autonomous parking," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 370–382, 2019.

[31] 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)*. IEEE, 2018, pp. 4327–4332.

[32] 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)*. IEEE, 2021, pp. 2927–2934.

[33] 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, 2022.

[34] R. Chai, A. Tsourdos, S. Chai, Y. Xia, A. Savvaris, and C. P. Chen, "Multiphase overtaking maneuver planning for autonomous ground vehicles via a desensitized trajectory optimization approach," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 74–87, 2022.

[35] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.

[36] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip, "Dynamically constrained motion planning networks for non-holonomic robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6937–6943.

[37] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia, and C. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1400–1413, 2020.

[38] R. Chai, H. Niu, J. Carrasco, F. Arvin, H. Yin, and B. Lennox, "Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[39] R. Chai, D. Liu, T. Liu, A. Tsourdos, Y. Xia, and S. Chai, "Deep learning-based trajectory planning and control for autonomous ground vehicle parking maneuver," *IEEE Transactions on Automation Science and Engineering*, 2022.

[40] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[41] M. da Silva 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, 2019.

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

[43] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," in *Proceedings of the 1995 ASME International Congress and Exposition*, 1995.

[44] I. Bae, J. Moon, J. Jhung, H. Suk, T. Kim, H. Park, J. Cha, J. Kim, D. Kim, and S. Kim, "Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles," *arXiv preprint arXiv:2001.03908*, 2020.

[45] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu, and F. Gao, "Generating large convex polytopes directly on point clouds," *ArXiv*, vol. abs/2010.08744, 2020.

[46] D. Avis, K. Fukuda, and S. Picozzi, "On canonical representations of convex polyhedra," in *Mathematical Software*. World Scientific, 2002, pp. 350–360.

[47] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[48] S. Cameron and R. Culley, "Determining the minimum translational distance between two convex polyhedra," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, 1986, pp. 591–596.

[49] M. Lutz and T. Meurer, "Efficient formulation of collision avoidance constraints in optimization based trajectory planning and control," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2021, pp. 228–233.

[50] W. Tribbey, "Numerical recipes," *Software engineering notes.*, vol. 35, no. 6, 2010-11-27.

[51] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, pp. 1–10, 2022.

[52] J. Leslie, "On a differential structure for the group of diffeomorphisms," *Topology*, vol. 6, no. 2, pp. 263–271, 1967.

[53] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[54] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[55] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5681–5686.

[56] 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, 2006.
[57] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.
[58] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.
[59] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.

Zhichao Han received the B.Eng. degree in Automation from Zhejiang University, Hangzhou, China, in 2021. Currently, he is actively pursuing a Ph.D. degree in the Fast Lab at Zhejiang University, where he is under the supervision of Prof. Fei Gao.

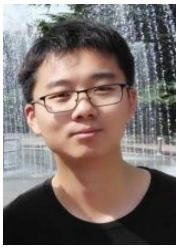
His research interests include motion planning and robot learning.

Yuwei Wu received a B.Eng. degree in transportation engineering from Beijing Jiaotong University, China, in 2019 and an M.SE degree in systems engineering from the University of Pennsylvania, in 2022. She is currently pursuing a Ph.D. degree in electrical and systems engineering with GRASP Laboratory, University of Pennsylvania.

Her research interest is in motion planning for autonomous vehicle systems.

Tong Li received his B.Eng. and M.Eng. in mechanical from Beijing Institute of Technology, China, in 2017 and 2019. He then joined HKUST Aerial Robotics Group at the Hong Kong University of Science and Technology, under the supervision of Prof. Shaojie Shen.

His research interests include prediction, decision-making, and motion planning for autonomous vehicles.



Lu Zhang received his B.Eng. and M.Eng. in automation from Beijing Institute of Technology, China, in 2015 and 2018. He then joined HKUST Aerial Robotics Group at the Hong Kong University of Science and Technology, under the supervision of Prof. Shaojie Shen.

His research interests cover decision-making, motion planning, and motion prediction for autonomous vehicles

Liuao Pei received the B.Eng. degree in detection, guidance and control technology from Harbin Institute of Technology, Harbin, China, in 2022. He is currently working toward the M.Phil. degree in control science and technology with Zhejiang University, Hangzhou, China.

His research interests include autonomous navigation and motion planning in complex environments.

Long Xu received the B.Eng. degree in automation from Zhejiang University, Hangzhou, China, in 2022. He is currently working toward the M.Phil. degree in electronic information with Zhejiang University, Hangzhou, China.

His research interests include ground robots, motion planning and robot learning.

Chengyang Li received the B.Eng. degree in computer science from the Southern University of Science and Technology. He is currently a Master of Philosophy (M.Phil.) student at the Hong Kong University of Science and Technology (Guangzhou).

His research interest includes robot localization, mapping, and motion planning.

Changjia Ma received the B.Eng. degree in electrical engineering and automation from Harbin Institute of Technology, Harbin, China, in 2021. He is currently working toward the MSc. degree in control science and engineering with Zhejiang University, Hangzhou, China.

His research interests include motion planning of autonomous vehicles and swarm robotics.

Chao Xu received his Ph.D. in Mechanical Engineering from Lehigh University in 2010. He is currently Associate Dean and Professor at the College of Control Science and Engineering, Zhejiang University. He serves as the inaugural Dean of ZJU Huzhou Institute, as well as plays the role of the Managing Editor for *IET Cyber-Systems & Robotics*.

His research expertise is Flying Robotics, Control-theoretic Learning. Prof. Xu has published over 100 papers in international journals, including *Science Robotics* (Cover Paper), *Nature Machine Intelligence* (Cover Paper), etc. Prof. Xu will join the organization committee of the IROS-2025 in Hangzhou.

Shaojie Shen received his B.Eng. degree in Electronic Engineering from the Hong Kong University of Science and Technology in 2009. He received his M.S. in Robotics and Ph.D. in Electrical and Systems Engineering in 2011 and 2014, respectively, all from the University of Pennsylvania, PA, USA.

He joined the Department of Electronic and Computer Engineering at the HKUST in September 2014 as an Assistant Professor, and was promoted to Associate Professor in July 2020. His research interests are in the areas of robotics and unmanned aerial vehicles, with focus on state estimation, sensor fusion, computer vision, localization and mapping, and autonomous navigation in complex environments.

Fei Gao received the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2019.

He is currently a tenured associate professor at the Department of Control Science and Engineering, Zhejiang University, where he leads the Flying Autonomous Robotics (FAR) group affiliated with the Field Autonomous System and Computing (FAST) Laboratory. His research interests include aerial robots, autonomous navigation, motion planning, optimization, and localization and mapping.