Trajectory Generation for Vehicle with Stable Time

***Abstract*—To do**

***Index Terms*—Motion Planning, Trajectory Optimization.**

I. Introduction

In the unfolding narrative of technological progress, the refinement of on-board computational systems has seamlessly woven mobile robotics into the variegated fabric of human enterprise and communal existence. Within the autonomous systems domain, motion planning has ascended to a critical role, incessantly drawing significant scholarly and industrial interest. However, the labyrinthine nature of complex environments coupled with the constraints of nonholonomic dynamics presents formidable impediments to the efficient forging of high-caliber trajectories. Additionally, a crucial, sought-after attribute in real-world deployment is the temporal stability of motion planning—consistent computational time across varied scenarios, which unmistakably raises the bar for algorithmic sophistication.

Conventionally, the objective of motion planning is to produce a collision-free and dynamically feasible trajectory from defined initial and final states of the robot. Traditional motion planning techniques are generally categorized into two classes: sampling-based and optimization-based methods. Sampling-based methods [1]–[4] are recognized for their resolution completeness and capability to evade local optima by generating and evaluating cost functions of samples within a discretized state space. Despite their merits, these methods face a combinatorial explosion when tasked with delivering high-quality trajectories, due to the increased dimensions of the state space. Optimization-based methods [5]–[8], in contrast, frame motion planning as a nonlinear optimal control problem within a continuous domain, utilizing numerical optimization algorithms to obtain locally optimal solutions. Although they hold the potential for swift generation of high-quality solutions, their success is significantly contingent upon the quality of the initial values, with subpar initial conditions possibly leading to infeasible local optima. To circumvent these limitations, a hierarchical planning framework [9]–[13] that amalgamates the advantages of both methods appears to be a rational solution. Within this structure, we define the front end of the framework as path planning while the back end focuses on trajectory optimization. The front end, always operating through sampling in a lower-dimensional configuration space, outputs a rough curve that typically omits a meticulous kinematic model and nuanced temporal planning, rendering it not directly amenable to low-level controllers. Additionally, in many case, the front-end output is not guaranteed to be collision-free. Subsequently, the back end takes this preliminary front-end output as the initial guess and meticulously optimizes it into a high-quality, safe and time-parameterized trajectory, dutifully considering the system's higher-order kinematic constraints.

Despite the streamlining and advancements offered by this hierarchical framework, it faces a triad of challenges when applied in complex environments. These challenges are the assurance of high-quality output, the stability of planning times, and the efficiency of the computational process. For the front end, conventional search or sampling algorithms may underperform in densely obstructed environments due to an incomplete understanding of the overall scene, leading to an abundance of non-contributory samples and excessive demands on computational and memory resources. This can adversely affect the consistency of planning times, a vital factor in real-time applications. For the back end, some methods depend excessively on prescriptive guidelines [14]–[16], circumscribing their adaptability to intricate and heterogenous scenarios. Owing to the complexities of nonholonomic kinematics, most of strategies mandate a markedly simplified motion paradigm [17, 18] or discrete motion process [9, 19, 20] to uphold algorithmic efficiency. Nonetheless, to ensure elevated executability and high success probabilities in dense settings, these strategies necessitate refined discretization, thereby adversely negatively impact the temporal efficiency of the planning process.

To address the aforementioned challenges, we have made significant improvements to both the front end and back end, ultimately integrating them into a novel learning-enhanced hierarchical planning framework. We design the front end as a lightweight two-stage neural network that takes the environment and encoded $\mathbb{SE}(2)$ states of the start and goal as inputs and efficiently outputs a path with high time stability. This network initially learns a rough guidance region at the global layer and then utilizes the local layer to further improve the path quality and achieve obstacle avoidance. In contrast to recent algorithms that integrate neural networks with sampling or searching techniques, our method stands out by directly outputting a path from the network, thus eliminating the need for additional sampling or searching and consequently increasing efficiency and temporal stability. By leveraging the direct use of ground truth trajectories, our front end more effectively emulates the behavior of the back-end planner, resulting in paths that are more amenable to optimization than those produced by above methods. For the back end, as in our previous work [13], we model it as a spatial-temporal joint trajectory optimization problem within a flat space, where all feasibility constraints are analytically represented by the optimized variables. Although flatness-based trajectory representations simplify the trajectory planning problem and enhance computational efficiency, they introduce singularities when the robot's velocity is zero, which can lead to discontinuities in motion or constraints that cannot be strictly satisfied. Thus, diverging from our prior work [13], this research re-characterizes the flat model and correspondingly designs a unique dual-layer piece-wise polynomial representation for

the trajectory. This new approach inherits the efficiency of the previous work while fundamentally addressing the issues of ambiguity, enhancing the algorithm's completeness and robustness. Additionally, to minimize mode switching between forward and backward movements of the vehicle, we propose an iterative modal planning algorithm which can effectively reduce unnecessary transitions. We conduct comprehensive ablation experiments and benchmarks to validate the effectiveness and advanced performance of our algorithm. Besides, real-world experiments also verify the practicality of our algorithm on a physical robot. Consequently, the main contributions of this paper can be summarized as follows:

- We propose a lightweight two-stage neural network that is capable of learning a path connecting the start and goal $\mathbb{SE}(2)$ states directly from the environment. A salient feature of our algorithm is its independence from traditional sampling or search algorithms, which significantly enhances time efficiency and temporal stability. Furthermore, our network is adept at learning the paradigms of ground truth trajectories, thereby providing superior initial path for optimization. This capability significantly reduces the back-end computational load and required optimization time.
- We devise a distinctive trajectory formulation characterized by a dual-layer, piece-wise polynomial architecture, upon which we construct a differential flatness-based spatial-temporal joint trajectory optimization formulation. This methodology is adept at not only facilitating the efficient generation of superior, collision-free trajectories but also at addressing the intrinsic ambiguities associated with flatness models, thus augmenting the completeness. Furthermore, we meticulously derive the gradient propagation chain within this trajectory representation, which empowers the proficient employment of gradient-based numerical optimization techniques.
- We amalgamate the proposed front-end and back-end components to present a high-quality vehicle trajectory planner. Moreover, we employ an iterative modal planning algorithm to efficaciously mitigate the transitions between forward and reverse motions during vehicle maneuvering. Additionally, we conduct a series of ablation studies to meticulously analyze the contribution of each component, and comparative experiments are undertaken to validate the superiority of our algorithm.

II. Related Work

A. Path Planning

Search or sampling-based methods [3, 21]–[25] have established a wide application in robotic motion planning, acclaimed for their potent generalization across various robotic systems and their flexibility in assimilating diverse user-specified constraints and objectives. Search-based strategies systematically discretize the state space, incrementally expanding a search tree by administering control inputs to individual nodes. Dolgov et al. [22] advance the hybrid A* algorithm, specifically devised for the kinematics of vehicles. Phillips et al. [21] pioneer the concept of safe interval path planning, effectively partitioning the temporal domain into viable intervals to circumvent dynamic obstacles. Ren et al. [23] build upon this foundation by incorporating multi-objective search techniques to facilitate multi-objective path planning challenges.Represented by rapidly-exploring random tree (RRT), Probabilistic planners [3, 24, 25] typically undertakes stochastic sampling within the configuration space, nurturing a growth tree from the starting point to obtain a feasible path. Karaman and Frazzoli [3] unveil a rewire mechanism, substantiating the inherent potential of RRT* for global optimality. Gammell et al. [24] introduce the concept of informed RRT*, amalgamating informed sampling strategies and heuristic guidance to boost efficiency. To address nonholonomic kinematics, Webb and Berg [25] expand the tree structure by sampling and connecting states in both configuration and control spaces.

However, due to intricate kinematics and a deficiency in global environmental comprehension, traditional search or sampling methods tend to proliferate superfluous samples in complex scenes, thereby imposing burdens on memory and computational efficiency. Recently, learning-based approaches [26]–[33] try to ameliorate this issue by leveraging neural networks to guide the search or sampling processes. Yonetani et al. [26] propose the neural A* framework, which is pivotal in modeling the conventional A* search as a differentiable process, thereby allowing it to be integrated into the training of neural networks. Dongchan Kim and Kunsoo Huh [29] develop a neural network for autonomous parking that predicts path distributions, which can then be employed to construct heuristic functions for search algorithms. Furthermore, Qureshi et al. [30] present the well-acclaimed Motion Planning Network (MPNet), which includes an Environment Network (Enet) and a Planning Network (Pnet). Enet is tasked with extracting environmental features, which Pnet then utilizes to determine the subsequent candidate state for sampling. Expanding on MPNet, Li et al. [32] incorporate nonholonomic kinematic constraints and deploy a discriminator network to select the most favorable candidate from a set of options produced by a generative network, thereby elevating the quality of the solution. Johnson et al. [33] harness a transformer model [34] to more adeptly extract features relevant to path planning problems, which ultimately generates a probabilistic map of sample distributions to provide additional information gain for searching or sampling. Admittedly, the aforementioned approaches substantially mitigate some of the shortcomings inherent in traditional methods. However, we contend that they do not fully exploit the capabilities of the networks. These methods still rely on search or sampling strategies, which compromises their temporal efficiency and stability. Furthermore, due to sampling in low-dimensional spaces, the objective functions at the sampling process often differ from those at the back end, resulting in paths that may not be conducive to back-end optimization. In contrast, our algorithm is more direct, obviating the need for additional search or sampling. By directly emulating expert trajectories, our approach narrows the gap between the front end and the back end, facilitating more efficient optimization in the latter stages.

B. Trajectory Optimization

Many studies [35]–[38] intuitively decouple space and time to simplify the trajectory optimization problem. The convex elastic band smoothing algorithm [35] circumvents the non-convexity of curvature constraints by assuming trajectory length consistency throughout each iteration. Zhou et al. [38] employ sequential convex optimization to relax curvature constraints for optimizing spatial shapes and propose piecewise-jerk speed optimization for temporal planning to minimize energy consumption. However, due to the decoupling of space and time, the aforementioned methods squander a considerable portion of the solution space, compromising optimality. Robust model predictive control approaches are often utilized to model the spatial-temporal trajectory optimization. Such methods [9, 10, 19, 39]–[42] discretize the motion process and ultimately model trajectory optimization as a nonlinear optimal control problem in the state space. In the work [19], a unified optimal control problem formulation is delineated for trajectory generation amidst unstructured environments with arbitrarily located obstacles, albeit with the caveat that the collision-free constraint is inherently non-differentiable. Zhang et al. [41] assume the convexity of obstacles and introduce dual variables to effectively eliminate the integer aspect of the original optimization problem, thereby reformulating the obstacle avoidance constraints into a differentiable form. He et al. [10] advance this paradigm [41] by introducing an efficient warm-start technique for the dual variables to accelerate optimization. Li et al. [9] propose an iterative algorithm with an inner loop that solves trajectory optimization subject to corridor constraints and an outer loop responsible for corridor rectification. Further, Sun et al. [42] present a successive linearization in feasible set algorithm, concentrating on the simplification of non-convex collision avoidance constraints and the linearization of nonlinear dynamic constraints proximal to the current trajectory.

The discretization of motion in trajectory planning often faces a tradeoff between quality and computational time, particularly in dense environments where higher discretization raises optimization dimensions, affecting time efficiency. Recently, Han et al. [13] model fully differentiable trajectory optimization in flat spaces with refined trajectory representations, validated for efficiency by extensive experiments. However, singularities at zero robot velocity challenge constraint satisfaction. While the work [43] introduce additional variables to resolve singularities, it is not a general-purpose trajectory planner. It only computes an optimal spline curve between the start and goal, which not only lacks the freedom to accommodate various scenarios, but also fails to address obstacle avoidance and dynamical constraints. Our algorithm synthesizes the above advances, fundamentally eliminating singularities and efficiently generating high-quality, collision-free trajectories across complex environments.

III. Learning-Based Path Planning

In this section, we introduce the learning-based path planner that leverages the performance of the network to directly output an initial path for optimization without the need for additional sampling or searching. Our path planner takes as input the navigation start $\boldsymbol{x}_i^p$ and goal $\boldsymbol{x}_N^p$, as well as the grid-based environment $\mathcal{E}$ of size $H*W$, and directly outputs a path $\mathbf{p} = \{\boldsymbol{x}_1^p, ..., \boldsymbol{x}_i^p, ..., \boldsymbol{x}_N^p\}$ serialized as a sequence of multiple state points. Here, the resolution of each grid is defined as r_s. Moreover, each state point in the path is associated with the $\mathbb{SE}(2)$ state of the robot, which includes its position and orientation angle. We firstly describe the network architecture, which initially estimates the distribution of each state point in the environmental space, and then further reduces their uncertainty to determine the precise states of the points in the sequence. Then, we discuss the loss function used during training, which includes supervising the network's output with the ground truth and penalizing infeasible paths based on safety considerations and nonholonomic dynamic constraints.

A. Neural Network Architecture

Our network comprises three main components: the feature extraction layer, the global distribution layer, and the local correction layer, as illustrated in Fig. 1. In practice, directly localizing the position of a specific state point within the environment is challenging and labor-intensive. Therefore, drawing inspiration from the R-CNN [44], our network follows a two-stage inference architecture. Initially, we uniformly partition $\mathcal{E}$ into $H_l * W_l$ region proposals, where each region proposal corresponds to a block of size $\frac{H}{H_l} * \frac{W}{W_l}$ in the environment. The center point of each region proposal is designated as an anchor point. Subsequently, we employ the global distribution layer to obtain the probability distribution of each state point with respect to the region proposals. This step involves estimating the coarse spatial distribution of the point. Next, leveraging the local correction layer, we further obtain the accurate position of the state point based on the environmental features and the probability distribution obtained earlier. Intuitively, our aim is to first approximately determine the region proposal in which each state point resides and then regress its positional offset relative to the anchor, thereby recovering the global position of the point.

The input to our network consists of the environment $\mathcal{E}$, the encoding of the start and goal states. Here, $\mathcal{E}$ is represented by a Euclidean Signed Distance Field (ESDF), where each element represents the signed distance from obstacles at that location. Inspired by the work [33], we adopt a start-goal encoding strategy by highlighting patches of size $c * c$ on a tensor of size $H * W$. Specifically, we assign a value of -1 to the patch representing the start point and a value of 1 to the patch representing the goal point. The remaining positions in the tensor are set as 0. To fully represent the $\mathbb{SE}(2)$ space, we introduce two additional $H * W$ tensors to capture the cosine and sine values of the robot orientations at the start and goal locations. Specifically, one tensor represents the cosine values for the patches corresponding to the starting and target points, while the other tensor represents the sine values. Subsequently, the aforementioned representations of the environment and the start and goal $\mathbb{SE}(2)$ states are concatenated to form a $4*H*W$ tensor, which serves as the input to the feature extraction layer of our model. The feature extraction layer consists of

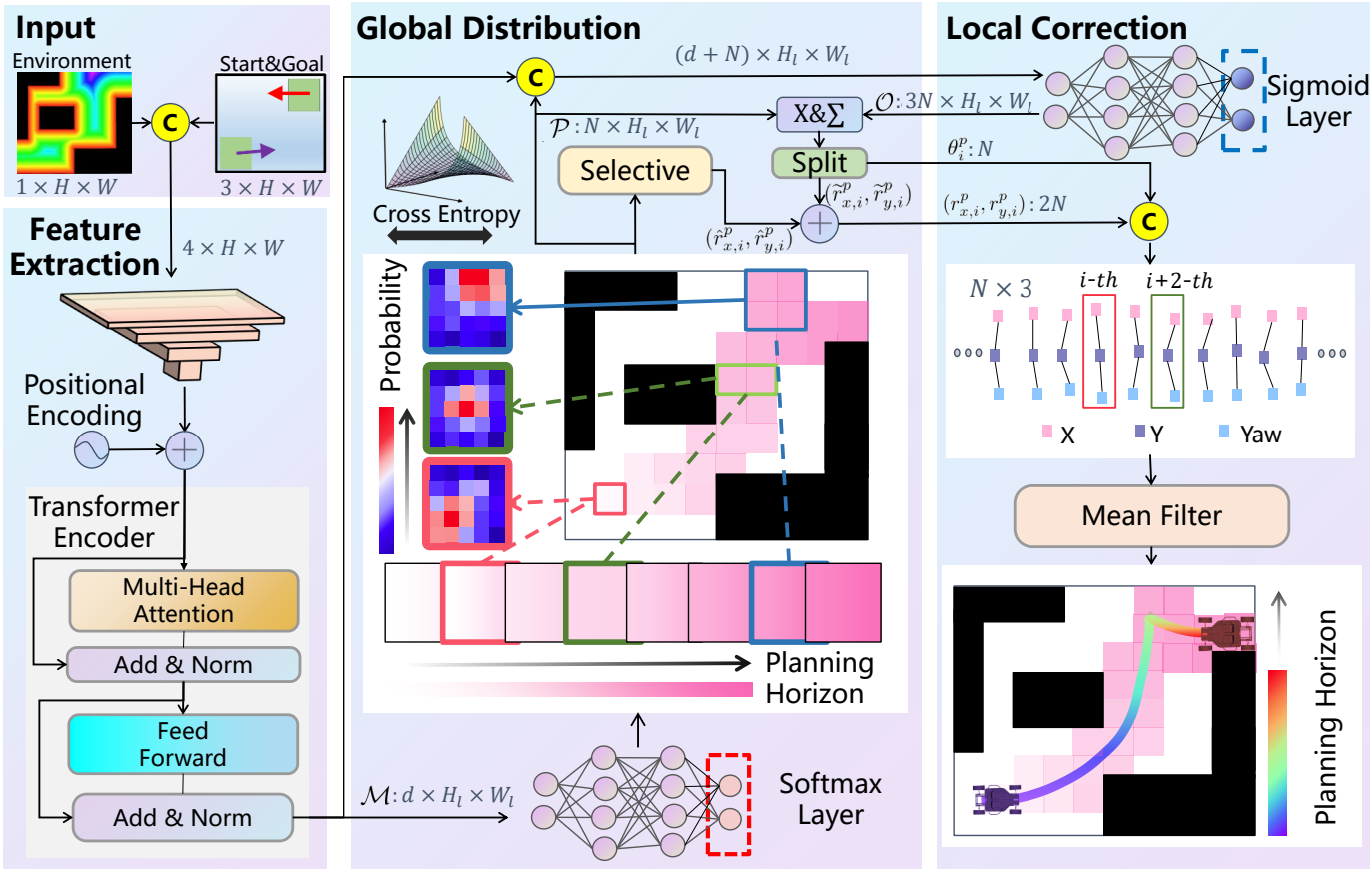

Fig. 1: Neural network visualization. The yellow circles containing the character **C** represent concatenation operators. The selection module implies the weighted computation of anchor positions based on the probability distribution $\mathcal{P}$.

a Fully Convolutional Network (FCN) and a Transformer Encoder [34]. The FCN encodes path plan problem into a high-dimensional latent space, and the size of this latent feature is downsampled to $d * H_l * W_l$ to match the shape of the probability distribution of region proposals. Here, d represents the user-defined dimension of the latent features. Next, we apply the transformer module to further fuse the features within the latent space, without altering the shape of the feature tensor. For brevity, we denote the output of the feature extraction layer as $\mathcal{M}$. $\mathcal{M}$ is further inputted to the global distribution layer, resulting in the probability distribution map $\mathcal{P}$ over the region proposals for N points along the path, with a size of $N*H_l*W_l$. Here, we denote the probability of the i-th point belonging to the (j,k)-th region as $\varrho_{i,j,k}$. Moreover, it is evident that these probabilities should satisfy the principle of probability normalization:

$$\sum_{j=0}^{H_l-1}\sum_{k=0}^{W_l-1} \varrho_{i,j,k} = 1, \forall i \in \{1, ..., N\}. \tag{1}$$

The probability distribution $\mathcal{P}$ and latent features $\mathcal{M}$ are concatenated and fed into the local correction layer, which outputs a tensor $\mathcal{O}$ of size $3N*H_l*W_l$. The physical meaning of this tensor $\mathcal{O}$ is to assign an orientation angle $b^\theta_{i,j,k}$ and positional offsets $(b^x_{i,j,k}, b^y_{i,j,k})$ relative to the anchor for each region. Instinctively, we consider selecting the region with the highest probability from $\mathcal{P}$ and retrieving the corresponding positional offset and orientation angle from $\mathcal{O}$ to accurately recover the $\mathbb{SE}(2)$ state of any point along the path. However, the operation of selecting the region with the highest probability is non-differentiable, posing challenges for training. Consequently, we employ a weighted summation based on $\mathcal{P}$ to compute the state of any point, enabling differentiability and facilitating effective training:

$$r^p_{x,i} = \hat{r}^p_{x,i} + \tilde{r}^p_{x,i}, \tag{2}$$

$$\hat{r}^p_{x,i} = \sum_{j=0}^{H_l-1}\sum_{k=0}^{W_l-1} \varrho_{i,j,k}(j+0.5)r_s, \tilde{r}^p_{x,i} = \sum_{j=0}^{H_l-1}\sum_{k=0}^{W_l-1} \varrho_{i,j,k} b^x_{i,j,k},$$

$$r^p_{y,i} = \hat{r}^p_{y,i} + \tilde{r}^p_{y,i}, \tag{3}$$

$$\hat{r}^p_{y,i} = \sum_{j=0}^{H_l-1}\sum_{k=0}^{W_l-1} \varrho_{i,j,k}(k+0.5)r_s, \tilde{r}^p_{y,i} = \sum_{j=0}^{H_l-1}\sum_{k=0}^{W_l-1} \varrho_{i,j,k} b^y_{i,j,k},$$

$$\theta^p_i = \sum_{j=0}^{H_l-1}\sum_{k=0}^{W_l-1} \varrho_{i,j,k} b^\theta_{i,j,k}, \forall i \in \{1, ..., N\}, \tag{4}$$

where $r^p_{x,i}, r^p_{y,i}$, and θ^p_i represent the position and orientation angle of the i-th state point, respectively. $\hat{r}^p_{x,i}$ and $\hat{r}^p_{y,i}$ represent the weighted sum of anchor positions based on the probability distribution $\mathcal{P}$, which can also be regarded as the output of the global distribution layer. For the sake of simplicity, we define the rough path composed of $(\hat{r}^p_{x,i}, \hat{r}^p_{y,i})$ as $\hat{\mathbf{P}}$. On the other hand, $\tilde{r}^p_{x,i}$ and $\tilde{r}^p_{y,i}$ refer to the weighted sum of positional offsets based on $\mathcal{P}$, which aims to correct $\hat{\mathbf{P}}$ and enhance its overall quality. Moreover, in order to mitigate outliers and

increase smoothness, we employ a sliding window for mean filtering along the aforementioned path. $(\hat{r}^p_{x,i}, \hat{r}^p_{y,i})$ $(\widetilde{r}^p_{x,i}, \widetilde{r}^p_{y,i})$ $(r^p_{x,i}, r^p_{y,i})$ Next, we provide the implementation details of each major part of the network.

1) **Feature Extraction**: In this component, the FCN consists of four dual-convolution layers. The first dual-convolution layer does not alter the width and height of the tensor, but only increases the number of channels. The subsequent three dual-convolution layers downsample the tensor in width and height, while doubling the number of channels after each layer. Moreover, MaxPool is employed for downsampling, and ReLU and Batch Normalization are nested between convolutions to enhance the generalization performance. Then, the output of the FCN is reshaped row-wise into a size of $d*(H_lW_l)$ and undergoes positional encoding before being fed into the Transformer Encoder. The Transformer module further learns the relationships between these latent features through multi-headed self-attention and multilayer perceptron blocks.
2) **Global Distribution**: This module predicts the probability distribution of N state points in the downsampled space, hence the output tensor dimension should be $N * H_l * W_l$. For this purpose, we efficiently implement this operation using two layers of $3*3$ padding convolutions. Additionally, we perform Softmax on each channel of the output tensor to ensure proper probability normalization.
3) **Local Correction**: Similar to the previous module, we employ two layers of $3*3$ padding convolutions to assign the orientation angle and position offsets relative to the anchor for each region. The output tensor has dimensions of $3N * H_l * W_l$. Furthermore, we apply the Sigmoid activation function to limit the position offsets of the network output, ensuring that the corrected point remains in the vicinity of the anchor.

B. Loss Function

We assume that the ground truth for each path planning problem has been pre-processed offline, with the method of their acquisition to be expounded upon in Sect.V. We desire to plan a path that can learn the behavior policy of the ground truth while satisfying fundamental constraints of nonholonomic motion and obstacle avoidance. Here, we present the key loss functions $\mathcal{L}_*$ used during training and denote the weight corresponding to each loss as w_*.

Anchor Point Classification Loss. We model the role of the global distribution layer as a multi-classification problem, where our objective is to maximize the probability of the region containing the ground truth point. We apply the cross-entropy loss to measure the discrepancy between the predicted region hypotheses and the ground truth:

$$\mathcal{L}_{ce} = -w_{ce} \sum_{i=1}^{N} \sum_{j=0}^{H_l-1} \sum_{k=0}^{W_l-1} \overline{\varrho}_{i,j,k} \log(\varrho_{i,j,k}). \tag{5}$$

where $\overline{\varrho}_{i,j,k}$ represents the probability ground truth.

Path Supervision Loss. We employ the Mean Square Error (MSE) loss to supervise the $\mathbb{SE}(2)$ state for each point along the path:

$$\begin{aligned}\mathcal{L}_{mse} = w_p \frac{\sum_{i=1}^{N}(r^p_{x,i} - \overline{r}^p_{x,i})^2 + (r^p_{y,i} - \overline{r}^p_{y,i})^2}{2N} \\ + w_\theta \frac{\sum_{i=1}^{N}(\cos\theta^p_i - \cos\overline{\theta}^p_i)^2 + (\sin\theta^p_i - \sin\overline{\theta}^p_i)^2}{2N},\end{aligned} \tag{6}$$

where $(\overline{r}^p_{x,i}, \overline{r}^p_{y,i}, \overline{\theta}^p_i)$ is the $\mathbb{SE}(2)$ state of the ground truth. Furthermore, to address potential periodicity issues with angles, we supervise the sine and cosine values of the angles instead of the angles themselves.

Smoothness Loss. In this part, we characterize smoothness by considering the path length and the total change in angle. Moreover, when N is sufficiently large, the difference between adjacent state points is small. Hence, to mitigate periodicity, we reasonably substitute the difference between the sines and cosines of angles for the angle difference. To emphasize the refinement of suboptimal cases where the planned path lacks smoothness compared to the ground truth, we design the following loss function to penalize such deviations:

$$\begin{aligned}\mathcal{L}_{smo} &= w_{arc}(\mathrm{ReLU}(\frac{\sum_{i=1}^{N-1}\Delta^p_{r,i}}{\sum_{i=1}^{N-1}\overline{\Delta}^p_{r,i}} - 1))^2 \\ &\quad + w_{rsm}(\mathrm{ReLU}(\frac{\sum_{i=1}^{N-1}\Delta^p_{\theta,i}}{\sum_{i=1}^{N-1}\overline{\Delta}^p_{\theta,i}} - 1))^2, \\ \Delta^p_{x,i} &= r^p_{x,i} - r^p_{x,i+1}, \Delta^p_{y,i} = r^p_{y,i} - r^p_{y,i+1}, \\ \Delta^p_{c,i} &= \cos\theta^p_i - \cos\theta^p_{i+1}, \Delta^p_{s,i} = \sin\theta^p_i - \sin\theta^p_{i+1}, \\ \Delta^p_{r,i} &= \sqrt{(\Delta^p_{x,i})^2 + (\Delta^p_{y,i})^2}, \Delta^p_{\theta,i} = \sqrt{(\Delta^p_{c,i})^2 + (\Delta^p_{s,i})^2},\end{aligned} \tag{7}$$

where $\Delta^p_{*,i}$ and $\overline{\Delta}^p_{*,i}$ denote the $\mathbb{SE}(2)$ state differences between adjacent points on the planned path and the ground truth, respectively.

Nonholonomic Dynamic Loss. We design the following loss function to emphasize the alignment between the direction of the line connecting adjacent points and the orientation angle direction:

$$\mathcal{L}_{hol} = \frac{w_{hol}}{N-1} \sum_{i=1}^{N-1} \mathrm{ReLU}((\left[\Delta^p_{x,i}, \Delta^p_{y,i}\right] \begin{bmatrix} -\sin\theta^p_{i+1} \\ \cos\theta^p_{i+1} \end{bmatrix})^2 - \delta_h), \tag{8}$$

where δ_h is the user-defined tolerance threshold.

Curvature Constraint Loss. We define the following loss function to penalize exceeding the specified curvature threshold κ_{max}:

$$\mathcal{L}_{cur} = \frac{w_{cur}}{N-1} \sum_{i=1}^{N-1} (\mathrm{ReLU}(\Delta^p_{\theta,i} - \kappa_{max} * \Delta^p_{r,i}))^2, \tag{9}$$

Uniform Loss. In the spatial domain, our objective is to attain a highly uniform distribution of points, which is achieved by penalizing variance. Moreover, we further normalize this variance by the path length of the ground truth, thereby ensuring that the network does not excessively prioritize long-

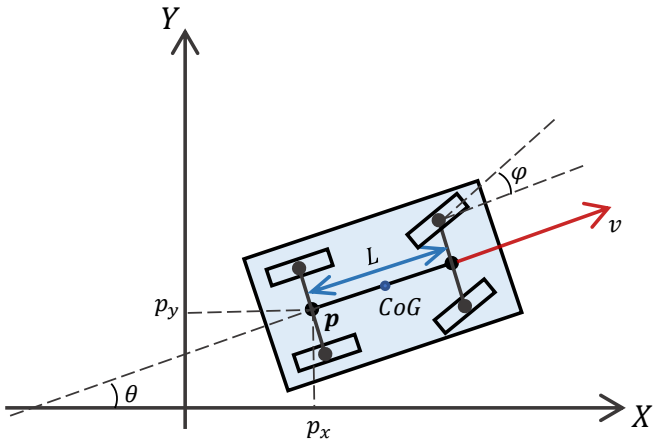

Fig. 2: The kinematic bicycle model.

distance path planning problems.

$$\mathcal{L}_{uni} = \frac{w_{uni}}{(N-1)\sum_{i=1}^{N-1}\Delta_{r,i}^{p}} \sum_{i=1}^{N-1}(\Delta_{r,i}^{p} - \frac{\sum_{j=1}^{N-1}\Delta_{r,j}^{p}}{N-1})^2, \tag{10}$$

Obstacle Avoidance Loss. We design the following loss function to emphasize the collision-free property of paths to the network:

$$\mathcal{L}_{obs} = w_{obs}\frac{\sum_{i=1}^{N}(\max(0, \delta_d - S_d(\boldsymbol{x}_1^p, \mathcal{S})))^2}{N}, \tag{11}$$

where S_d represents the signed distance of the state point from obstacles, taking into account the shape of the robot denoted by $\mathcal{S}$. $\delta_d \in \mathbb{R}^+$ is the minimum safety distance from obstacles set by the user. The signed distance at any grid can be obtained directly from the ESDF, and we utilize bilinear interpolation to smooth out the discontinuities caused arising from the discretization of space.

IV. Gradient-Based Trajectory Optimization

In this section, we conduct spatial-temporal optimization on the rough path obtained in the previous section, taking into account a higher-dimensional and more accurate dynamic model, while ensuring collision-free property. Firstly, we introduce the differential flatness model for car-like robots, which serves as a cornerstone for subsequent optimization modeling. Next, we present the trajectory optimization formulation in the flat space. In comparison to our previous work [13], one significant distinction lies in the design of a dual-layer polynomial-based trajectory parameterization. This approach inherits the efficient characteristics of the prior work while fundamentally addressing the singularity issues inherent in the flat model. Finally, we discribe the gradient backpropagation computation process and practical solution details specific to this optimization formulation.

A. Differential Flatness-Based Vehicle Model

We employ the simplified kinematic bicycle model to depict a four-wheel vehicle. Under the assumption of front-wheel drive and flawless rolling with no slippage, the motion model is represented as illustrated in Fig.2. Here, we consider a more detailed and accurate dynamic model: $(p_x, p_y, \theta, v, a, \phi, \omega)^{\mathrm{T}}$,

$$\frac{d}{dt}\begin{bmatrix} p_x \\ p_y \\ \theta \\ v \\ \phi \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ v\tan\phi/L \\ a \\ \omega \end{bmatrix}, \tag{12}$$

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 represents the longitude acceleration, ϕ is the steering angle of the front wheels, ω is the steer angular velocity, and L is the wheelbase. Subsequently, we select the flat output $\boldsymbol{\sigma} := (\sigma_x, \sigma_y)^{\mathrm{T}}$, where $\boldsymbol{\sigma} = \boldsymbol{p}$ represents the position, notably centered on the rear wheel of the vehicle. For computational convenience, we introduce an auxiliary antisymmetric matrix $\mathbf{B} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and the transformation of the remaining motion variables can be described as follows:

$$v = \eta||\dot{\boldsymbol{\sigma}}_{|t}||_2, \theta = \arctan 2(\eta\dot{\sigma}_{y|t}, \eta\dot{\sigma}_{x|t}), \tag{13a}$$

$$a = \eta\frac{\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}}\dot{\boldsymbol{\sigma}}_{|t}}{||\dot{\boldsymbol{\sigma}}_{|t}||_2} \tag{13b}$$

$$\phi = \arctan\left(\eta\frac{\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\sigma}}_{|t}L}{||\dot{\boldsymbol{\sigma}}_{|t}||_2^3}\right), \tag{13c}$$

$$\omega = \eta L\frac{\dddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\sigma}}_{|t}||\dot{\boldsymbol{\sigma}}_{|t}||_2^3 - 3\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\sigma}}_{|t}\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}}\dot{\boldsymbol{\sigma}}_{|t}||\dot{\boldsymbol{\sigma}}_{|t}||_2}{||\dot{\boldsymbol{\sigma}}_{|t}||_2^6 + (\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\sigma}}_{|t}L)^2}. \tag{13d}$$

$\dot{\boldsymbol{\sigma}}_{|t}, \ddot{\boldsymbol{\sigma}}_{|t}$ and $\dddot{\boldsymbol{\sigma}}_{|t}$ are the first, second, and third derivatives of the flat output w.r.t time, respectively. $\eta \in \{-1, 1\}$ is an additional variable fixed in optimization to characterize the motion direction of the vehicle, where $\eta = -1$ and $\eta = 1$ represent the backward and forward movements, respectively. Hence, leveraging the inherent differential flatness property, we can employ the flat outputs and their finite derivatives to represent various state variables of the robot, thereby streamlining trajectory planning and enhancing optimization processes.

B. Dual-Layer Polynomial-Based Optimization Formulation

In our previous work [13], we parametrized the flat output $\boldsymbol{\sigma}$ directly as a piecewise polynomial function of time. While this method is efficient and straightforward, Eq. (13c) and Eq. (13d) face singularities when the robot's velocity $\dot{\boldsymbol{\sigma}}_{|t}$ is zero. This led to numerical instability in the optimization process, making it difficult to guarantee strict satisfaction of constraints. By contrast, in this paper, we intuitively introduce a *Pseudo Arc* parametrized in the same piece-wise polynomial fashion. By indirectly deriving the state variables using the flat model, this approach fundamentally eradicates the singularity issues:

$$\boldsymbol{\sigma} = \boldsymbol{\gamma}(s), s = s(t). \tag{14}$$

Here, $s \in \mathbb{R}^+$ is the *Pseudo Arc* and we refer to this indirect trajectory representation method as the Dual-Layer Polynomial-Based parametrization. Then the finite-

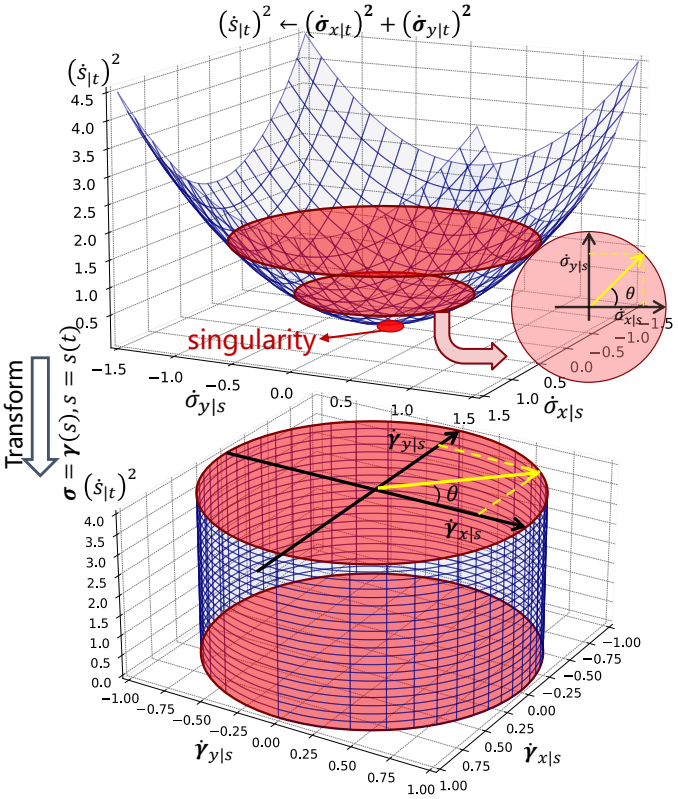

Fig. 3: The reparameterized visualization.

dimensional derivative of the flat output can also be derived as follows:

$$\dot{\boldsymbol{\sigma}}_{|t} = \dot{\boldsymbol{\gamma}}_{|s}\dot{s}_{|t}, \tag{15a}$$

$$\ddot{\boldsymbol{\sigma}}_{|t} = \dot{\boldsymbol{\gamma}}_{|s}\ddot{s}_{|t} + \dot{s}^2_{|t}\ddot{\boldsymbol{\gamma}}_{|s}, \tag{15b}$$

$$\dddot{\boldsymbol{\sigma}}_{|t} = \dddot{s}_{|t}\dot{\boldsymbol{\gamma}}_{|s} + 3\dot{s}_{|t}\ddot{\boldsymbol{\gamma}}_{|s}\ddot{s}_{|t} + \dot{s}^3_{|t}\dddot{\boldsymbol{\gamma}}_{|s}, \tag{15c}$$

$\dot{\boldsymbol{\gamma}}_{|s}, \ddot{\boldsymbol{\gamma}}_{|s}$ and $\dddot{\boldsymbol{\gamma}}_{|s}$ are the first, second, and third derivatives of the flat output w.r.t *Pseudo Arc*, respectively. It is worth mentioning that $\dot{\boldsymbol{\sigma}}_{|t}$ represents the actual motion velocity of the robot, while $\dot{\boldsymbol{\gamma}}_{|s}$ is defined as the *Pseudo Velocity*. Moreover, the physical meaning of $\dot{s}_{|t} \geq 0$ is the magnitude of velocity along the *Pseudo Arc*. We substitute Eq. (15a-15c) into Eq. (13a-13d), thus modifying the differential flatness model as follows:

$$v(t) = \eta||\dot{\boldsymbol{\gamma}}_{|s}||_2\dot{s}_{|t}, \theta = \arctan 2(\eta\dot{\gamma}_{y|s}, \eta\dot{\gamma}_{x|s}), \tag{16a}$$

$$a(t) = \eta(\ddot{s}_{|t}||\dot{\boldsymbol{\gamma}}_{|s}||_2 + \frac{\dot{s}^2_{|t}\ddot{\boldsymbol{\gamma}}^{\mathrm{T}}_{|s}\dot{\boldsymbol{\gamma}}_{|s}}{||\dot{\boldsymbol{\gamma}}_{|s}||_2}), \tag{16b}$$

$$\phi(t) = \arctan\left(\eta\frac{\ddot{\boldsymbol{\gamma}}^{\mathrm{T}}_{|s}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}L}{||\dot{\boldsymbol{\gamma}}_{|s}||^3_2}\right), \tag{16c}$$

$$\omega(t) = \eta L\frac{\dddot{\boldsymbol{\gamma}}^{\mathrm{T}}_{|s}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}||\dot{\boldsymbol{\gamma}}_{|s}||^3_2 - 3\ddot{\boldsymbol{\gamma}}^{\mathrm{T}}_{|s}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}\ddot{\boldsymbol{\gamma}}^{\mathrm{T}}_{|s}\dot{\boldsymbol{\gamma}}_{|s}||\dot{\boldsymbol{\gamma}}_{|s}||_2}{||\dot{\boldsymbol{\gamma}}_{|s}||^6_2 + (\ddot{\boldsymbol{\gamma}}^{\mathrm{T}}_{|s}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}L)^2}\dot{s}_{|t}. \tag{16d}$$

When the robot's velocity is zero, we can set $\dot{s}_{|t}$ to zero while keeping *Pseudo Velocity* $\dot{\boldsymbol{\gamma}}_{|s}$ non-zero. In other words, in principle, even during the transition between forward and backward motion states with zero velocity, we can ensure that $\dot{\boldsymbol{\gamma}}_{|s}$ remains non-zero. Consequently, the denominator of Eq. (16b-16d) are strictly positive, eliminating the original singularity point. Next, we provide a detailed exposition of the parametric form of $\boldsymbol{\sigma}(t) = \boldsymbol{\gamma}(s(t))$ based on dual-layer piece-wise polynomials.

Based on the front-end output, we first segment the trajectory according to the forward and backward modes. The i-th trajectory segment $\boldsymbol{\gamma}_i(s)$ is formulated as a M_i-piece polynomial with degree $D = 2u-1$, which is parameterized by the *Pseudo Arc* $\delta\boldsymbol{s}_i = (\delta s_{i,1}, ..., \delta s_{i,M_i})^{\mathrm{T}} \in \mathbb{R}^{M_i}$ corresponding to each piece and the coefficient matrix $\boldsymbol{c}^p_i = \left((\boldsymbol{c}^p_{i,1})^{\mathrm{T}}, ..., (\boldsymbol{c}^p_{i,M_i})^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2M_iu\times 2}$. Similarly, the i-th *Pseudo Arc* $s_i(t)$ is represented as a one-dimensional and time-uniform M_i-piece polynomial, parameterized by the time interval for each piece δT_i and coefficient matrix $\boldsymbol{c}^s_i = \left((\boldsymbol{c}^s_{i,1})^{\mathrm{T}}, ..., (\boldsymbol{c}^s_{i,M_i})^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2M_iu}$. Moreover, we consider a strict correspondence between each piece of the dual-layer piece-wise polynomials, such that for every time interval δT_i, the robot should traverse the corresponding *Pseudo Arc*:

$$\begin{aligned} s_i(j*\delta T_i) &= \sum_{o=1}^{j}\delta s_{i,o}, \\ \forall i \in \{1,2,...,n\}, &\forall j \in \{1,2,3,...,M_i\}, \end{aligned} \tag{17}$$

where n is the number of trajectory segments. Based on the above modeling, the j-th piece of the i-th segment $\boldsymbol{\gamma}_{i,j}$ is represented as:

$$\begin{aligned} \boldsymbol{\gamma}_{i,j}(s_{i,j}) &:= (\boldsymbol{c}^p_{i,j})^{\mathrm{T}}\boldsymbol{\beta}(s_{i,j} - \sum_{o=1}^{j-1}\delta s_{i,o}), \\ s_{i,j}(t) &:= (\boldsymbol{c}^s_{i,j})^{\mathrm{T}}\boldsymbol{\beta}(t), \forall t \in [0, \delta T_i], \\ \boldsymbol{\beta}(x) &:= \left(1, x, x^2, ..., x^N\right)^{\mathrm{T}}, \\ \forall i \in \{1,2,...,n\}, &\forall j \in \{1,2,3,...,M_i\}, \end{aligned} \tag{18}$$

where $\boldsymbol{\beta}(x)$ is a natural basis function. Moreover, due to the strict correspondence between each piece of the dual-layer polynomials, we can further derive the following equation from Eq. (17):

$$\begin{aligned} s_{i,j}(0) &:= \sum_{o=1}^{j-1}\delta s_{i,o}, s_{i,j}(\delta T_i) := s_{i,j}(0) + \delta s_{i,j}, \\ \forall i \in \{1,2,...,n\}, &\forall j \in \{1,2,3,...,M_i\}, \end{aligned} \tag{19}$$

where $s_{i,1}(0)$ is set as 0. Furtheremore, the M_i-piece polynomial $\boldsymbol{\gamma}_i$ is obtained:

$$\begin{aligned} \boldsymbol{\gamma}_i(s_i) =& \boldsymbol{\gamma}_{i,j}(s_i), s_i(t) = s_{i,j}(t - (j-1)*\delta T_i), \\ \forall j \in& \{1,2,...,M_i\}, t \in [(j-1)*\delta T_i, j*\delta T_i). \end{aligned} \tag{20}$$

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

$$\begin{aligned} \boldsymbol{\sigma}(t) &= \boldsymbol{\gamma}_i(s_i(t - \sum_{k=1}^{i-1}T_i)), \\ \forall i \in \{1,2,...,n\}, &t \in [\hat{T}_i, \hat{T}_{i+1}), \end{aligned} \tag{21}$$

where $T_s = \sum_{i=1}^{n} T_i$ represents the duration of the entire trajectory, $\hat{T}_i = \sum_{o=1}^{i-1} T_o$ denotes the timestamp of the starting point of the i-th segment, and $\hat{T}_1$ is set to 0. Moreover, for subsequent derivations, we define the sets of coefficient matrices for the dual-layer polynomials $\mathbf{c}^p = \left((\boldsymbol{c}_1^p)^{\mathrm{T}}, ..., (\boldsymbol{c}_n^p)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{\left(\sum_{i=1}^{n} 2M_i u\right)\times 2}$, $\mathbf{c}^s = \left((\boldsymbol{c}_1^s)^{\mathrm{T}}, ..., (\boldsymbol{c}_n^s)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{\sum_{i=1}^{n} 2M_i u}$, the *Pseudo Arc* set $\delta\mathbf{s} = \left(\delta\boldsymbol{s}_1^{\mathrm{T}}, ..., \delta\boldsymbol{s}_n^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{\sum_{i=1}^{n} M_i}$, , and the time set $\mathbf{T} = (T_1, ..., T_n)^{\mathrm{T}} \in \mathbb{R}^n$. With motion feasibility constraints, the minimum control effort problem based on the modified flatness model Eq. (16a-16d), incorporating first-order temporal regularization, is formulated as a nonlinear constrained optimization:

$$\min_{\mathbf{c}^p, \mathbf{c}^s, \delta\mathbf{s}, \mathbf{T}} J(\mathbf{c}^p, \mathbf{c}^s, \delta\mathbf{s}, \mathbf{T}) = \int_0^{T_s} \boldsymbol{\sigma}^{(u)}(t)^{\mathrm{T}} \boldsymbol{\sigma}^{(u)}(t) dt + \rho T_s \tag{22a}$$

$$\text{s.t.} \mathcal{B}_{ini}(\boldsymbol{\sigma}(0), ..., \boldsymbol{\sigma}^{(u-1)}(0)) = 0, \tag{22b}$$

$$\mathcal{B}_{fin}(\boldsymbol{\sigma}(T_s), ..., \boldsymbol{\sigma}^{(u-1)}(T_s)) = 0, \tag{22c}$$

$$\boldsymbol{\gamma}_{i,j}^{[\widetilde{d}]}(\sum_{k=1}^{j} \delta s_{i,k}) = \boldsymbol{\gamma}_{i,j+1}^{[\widetilde{d}]}(\sum_{k=1}^{j} \delta s_{i,k}), \tag{22d}$$

$$s_{i,j}^{[\widetilde{d}]}(\delta T_i) = s_{i,j+1}^{[\widetilde{d}]}(0), \tag{22e}$$

$$T_i > 0, \tag{22f}$$

$$||\dot{\boldsymbol{\gamma}}_{i|s}(s_i(t))||_2 > \alpha, \tag{22g}$$

$$s_i^{(1)}(t) \geq 0, \forall t \in [0, T_i], \tag{22h}$$

$$\mathcal{G}(\boldsymbol{\gamma}_i(s_i(t)), ..., \boldsymbol{\gamma}_{i|s}^{(u)}(s_i(t)), s_i(t), ..., s_{i|t}^{(u)}(t)) \preceq 0, \tag{22i}$$

$$\forall i \in \{1, ..., n\}, \forall j \in \{1, ..., M_i - 1\}, \forall t \in [0, T_i]. \tag{22j}$$

$u = 3$ is the control dimension and $\rho \in \mathbb{R}^+$ is a user-defined weight for the time regularization term to restrict the total duration T_s. Eq. (22b) and Eq. (22c) represent the initial and final state constraints of the trajectory, respectively. Eq. (22d)(22e) denotes the continuity constraints up to degree $\widetilde{d}$ at the junctions of the piece-wise polynomials. Eq. (22f) and Eq. (22h) correspond to positive-definite constraints on time and *Pseudo Velocity*, respectively. Eq. (22g) is a minimum *Pseudo Velocity* constraint introduced to avoid singularities, where α is a threshold value. $\mathcal{G}$ encompasses common inequality constraints considered in trajectory planning problems, including dynamic feasibility and obstacle avoidance constraints, the specific forms of which can be found in our previous work [13].

C. Gradient Back Propagation

We discretize each piece of the trajectory uniformly in time into $\lambda \in \mathbb{N}^+$ constraint points to approximate the continuous-time formulations Eq. (22a)(22i). For instance, feasibility constraints Eq. (22i) are transformed into their discrete counterparts as follows:

$$\mathcal{G}(\boldsymbol{\gamma}_i(s_i(t)), ..., \boldsymbol{\gamma}_{i|s}^{(u)}(s_i(t)), s_i(t), ..., s_{i|t}^{(u)}(t)) \preceq 0, \tag{23}$$
$$\forall i \in \{1, ..., n\}, \forall t \in [0, T_i],$$

$$\Rightarrow \mathcal{G}(\boldsymbol{\gamma}_{i,j,k}, ..., \boldsymbol{\gamma}_{i,j,k|s}^{(u)}, s_{i,j,k}, ..., s_{i,j,k|t}^{(u)}) \leq 0, \tag{24}$$

$$\boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d})} = \boldsymbol{\gamma}_{i,j|s}^{(\overline{d})}(s_{i,j,k}), s_{i,j,k|t}^{(\overline{d})} = s_{i,j|t}^{(\overline{d})}(\frac{kT_i}{\lambda M_i}), \tag{25}$$
$$\forall \overline{d} \in \{0, ..., u\}, \forall i \in \{1, ..., n\},$$
$$\forall j \in \{1, ..., M_i\}, \forall k \in \{0, ..., \lambda\}.$$

Our optimization modeling Eq. (22a-22j) is based on the differential flatness model of the robot Eq. (16a-16d), which implies that the loss function and constraints can be analytically determined by the variables $\boldsymbol{\gamma}$, s, and their finite-dimensional derivatives $\boldsymbol{\gamma}_{|s}^{(\overline{d})}, s_{|t}^{(\overline{d})}$. Hence, without loss of generality, we primarily derive the gradients of $s_{i,j,k|t}^{(\overline{d})}$ and $\boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d})}$ w.r.t the polynomial parameters at constraint points, while the remaining terms can be obtained sequentially by utilizing the chain rule:

$$\frac{\partial s_{i,j,k|t}^{(\overline{d})}}{\partial \boldsymbol{c}_{i,j}^s} = \boldsymbol{\beta}^{(\overline{d})}(\frac{kT_i}{\lambda M_i}), \frac{\partial s_{i,j,k|t}^{(\overline{d})}}{\partial T_i} = s_{i,j,k|t}^{(\overline{d}+1)} \frac{k}{\lambda M_i}, \tag{26}$$

$$\frac{\partial \boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d})}}{\partial \boldsymbol{c}_{i,j}^s} = \boldsymbol{\beta}(\frac{kT_i}{\lambda M_i})(\boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d}+1)})^{\mathrm{T}}, \tag{27}$$

$$\frac{\partial \boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d})}}{\partial T_i} = \boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d}+1)} \frac{k}{\lambda M_i} s_{i,j,k|t}^{(1)}, \tag{28}$$

$$\frac{\partial \boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d})}}{\partial \delta s_{i,o}} = \begin{cases} -\boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d}+1)} & 1 \leq o \leq j-1, \\ \mathbf{0} & j \leq o \leq M_i, \end{cases} \tag{29}$$

$$\frac{\partial \boldsymbol{\gamma}_{i,j,k|s}^{(\overline{d})}}{\partial \boldsymbol{c}_{i,j}^p} = \begin{bmatrix} \boldsymbol{\beta}^{(\overline{d})}(s_{i,j,k} - \sum_{o=1}^{j-1} \delta s_{i,o}), \mathbf{0} \\ \mathbf{0}, \boldsymbol{\beta}^{(\overline{d})}(s_{i,j,k} - \sum_{o=1}^{j-1} \delta s_{i,o}) \end{bmatrix}, \tag{30}$$

where $\mathbf{0}$ is a zero vector with the proper dimension.

D. Optimization Solution Details

We instantiate the initial and final state constraints Eq. (22b)(22c) for the robot, where the corresponding state is defined as $\boldsymbol{x} = \left(\boldsymbol{p}^{\mathrm{T}}, \theta, v, a, \phi\right)^{\mathrm{T}}$. Without loss of generality, taking the initial state constraint Eq. (22b) as an example, we utilize the flatness model Eq. (16a-16d) to convert it into boundary conditions applied to $\boldsymbol{\gamma}_1$ and s_1:

$$\boldsymbol{\gamma}_1(0) = \boldsymbol{p}, s_1(0) = 0, \tag{31}$$

$$\dot{\boldsymbol{\gamma}}_{1|s}(0) = (\eta\cos\theta, \eta\sin\theta)^{\mathrm{T}}, \dot{s}_{1|t}(0) = ||v||_2, \tag{32}$$

$$\ddot{\boldsymbol{\gamma}}_{1|s}(0) = (-\sin\theta\frac{\tan\phi}{L}, \cos\theta\frac{\tan\phi}{L}), \ddot{s}_{1|t}(0) = \frac{a}{\eta}. \tag{33}$$

Similarly, the final state constraint can also be transformed into the flatness space accordingly. Based on the established optimality condition in reference [45], the coefficients of a minimum control effort piece-wise polynomial can be analytically represented by the piece duration, connection points between adjacent pieces, as well as the head and tail states. Moreover, the propagation of gradients follows a linear com-

plexity. Hence, we can leverage the aforementioned principle to reparameterize the piece-wise polynomial, thereby ensuring the natural fulfillment of equality constraints, including boundary conditions Eq. (31-33) and continuity constraints Eq. (22d-22e). Moreover, we employ the Augmented Lagrange Multiplier method (ALM) [46] to relax inequality constraints such as Eq. (22i), which is a robust approach for addressing constrained optimization problems, involving the smoothing of the dual function through the addition of quadratic terms. This method iteratively solves the approximate unconstrained problem while updating the dual variables. To address the unconstrained optimization problem within the inner iteration, we utilize the L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) method [47], a powerful quasi-Newton optimization algorithm renowned for its effectiveness. By leveraging information from previous objective function values and gradients, the L-BFGS method estimates the Hessian matrix and effectively manages memory constraints. Furthermore, we analytically compute all gradients during implementation, which significantly enhances time efficiency and avoids numerical approximations.

E. Iterative Modal Planning

Thus far, we have successfully accomplished the modeling and solution of trajectory optimization. Nonetheless, the forward and backward characteristics of individual trajectory segments are determined by the coarse frontend, potentially leading to superfluous mode transitions. These transitions impose unwarranted expenditures of energy and time, while also contributing to tire wear. To tackle this challenge, we propose an iterative mechanism aimed at eliminating unnecessary mode switches, thereby facilitating the automatic search of optimal modal sequences, as depicted in Alg. 1. Initially, we employ the output of our path planning algorithm as the initial mode sequence, and proceed with trajectory optimization. Within each iteration, we efficiently examine the horizon of each trajectory segment and heuristically attempt to remove segments with a horizon T_i smaller than a predefined threshold ϵ. Furthermore, we merge the $(i-1)$-th and $(i+1)$-th trajectory segments since they inevitably belong to the same mode. Subsequently, the pruned and restructured trajectory serve as the initial guess for subsequent optimization. Besides, we subject the optimized trajectory to feasibility checks. In the event of an infeasible solution, indicating the absence of a viable solution under the current mode configuration, we revert to the feasible trajectory output from the previous iteration, even if it entails a higher frequency of transitions. Conversely, if the optimized trajectory satisfies the feasibility check, the iterative process continues until the horizon of each trajectory segment surpasses a predetermined threshold, signifying the necessity of every transition.

Algorithm 1: Iterative Trajectory Planning Framework

Input: environment: $\mathcal{E}$, initial state: $\boldsymbol{x}_{ini}$, final state: $\boldsymbol{x}_{fin}$, threshold: ϵ, maximum iteration: N_{iter}.
Output: optimal trajectory: ξ^o.

$\mathbf{p} \leftarrow$ `PathPlanning`$(\mathcal{E}, \boldsymbol{x}_{ini}, \boldsymbol{x}_{fin})$;
$\xi^o \leftarrow$ `TrajOptimization`$(\mathbf{p})$;
$m \leftarrow 1$;
while $m++ \leq N_{iter}$ **do**
  $\mathcal{V}$.`Clear()`;
  for $i \leftarrow 1$ **to** n **do**
    if $T_i \leq \epsilon$ **then**
      $\mathcal{V}$.`PushBack(`i`)`;
    end
  end
  $\xi \leftarrow$`PruneAndMerge(`$\xi^o, \mathcal{V}$`)`;
  $\xi \leftarrow$`TrajOptimization(`ξ`)`;
  if `NotFeasible(`$\xi, \mathcal{E}$`)` **then**
    break;
  else
    $\xi^o \leftarrow \xi$;
  end
end
return ξ^o

V. Evaluations

VI. Conclusion

References

[1] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model," in *2006 2nd International Conference on Information & Communication Technologies*, vol. 1. IEEE, 2006, pp. 781–786.

[2] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011, pp. 723–730.

[3] 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.

[4] C.-b. Moon and W. Chung, "Kinodynamic planner dual-tree rrt (dt-rrt) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Transactions on industrial electronics*, vol. 62, no. 2, pp. 1080–1090, 2014.

[5] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2009, pp. 489–494.

[6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[7] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2020.

[8] H. Liu, W. Dong, Z. Zhang, C. Wang, R. Li, and Y. Gao, "Optimization-based local planner for a nonholonomic autonomous mobile robot in semi-structured environments," *Robotics and Autonomous Systems*, p. 104565, 2023.

[9] 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.

[10] 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.

[11] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4327–4332.

[12] S. Zhang, Z. Jian, X. Deng, S. Chen, Z. Nan, and N. Zheng, "Hierarchical motion planning for autonomous driving in large-scale complex scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13 291–13 305, 2021.

[13] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen, and F. Gao, "An efficient spatial-temporal trajectory planner for

autonomous vehicles in unstructured environments," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–18, 2023.

[14] F. Gómez-Bravo, F. Cuesta, A. Ollero, and A. Viguria, "Continuous curvature path generation based on β-spline curves for parking manoeuvres," *Robotics and autonomous systems*, vol. 56, no. 4, pp. 360–372, 2008.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[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] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Chicago, IL, Sept 2014, pp. 2997–3004.

[25] 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.

[26] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path planning using neural a* search," in *International conference on machine learning*. PMLR, 2021, pp. 12 029–12 039.

[27] J. Huh, D. D. Lee, and V. Isler, "Learning continuous cost-to-go functions for non-holonomic systems," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5772–5779.

[28] J. Wang, X. Jia, T. Zhang, N. Ma, and M. Q.-H. Meng, "Deep neural network enhanced sampling-based path planning in 3d space," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3434–3443, 2021.

[29] D. Kim and K. Huh, "Neural motion planning for autonomous parking," *International Journal of Control, Automation and Systems*, vol. 21, no. 4, pp. 1309–1318, 2023.

[30] 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.

[31] 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.

[32] L. Li, Y. Miao, A. H. Qureshi, and M. C. Yip, "Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4496–4503, 2021.

[33] J. J. Johnson, U. S. Kalra, A. Bhatia, L. Li, A. H. Qureshi, and M. C. Yip, "Motion planning transformers: A motion planning framework for mobile robots," *arXiv preprint arXiv:2106.02791*, 2021.

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[35] 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.

[36] 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.

[37] X. Shi and X. Li, "Speed planning for an autonomous vehicle with conflict moving objects," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 413–418.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] C. Sun, Q. Li, B. Li, and L. Li, "A successive linearization in feasible set algorithm for vehicle motion planning in unstructured and low-speed scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3724–3736, 2021.

[43] R. Walambe, N. Agarwal, S. Kale, and V. Joshi, "Optimal trajectory generation for car-type mobile robot using spline interpolation," *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 601–606, 2016.

[44] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

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

[46] R. T. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming," *SIAM Journal on Control*, vol. 12, no. 2, pp. 268–285, 1974.

[47] 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.