Trajectory Generation for Vehicle with Stable Time

Zhichao Han[1,2], Mengze Tian[1,2] and Fei Gao[*,1,2]

[1]Institute of Cyber-Systems and Control,
College of Control Science and Engineering,
Zhejiang University, Hangzhou 310027, China.
[2]Huzhou Institute of Zhejiang University, Huzhou 313000, China.
[*] Corresponding author. Email: fgaoaa@zju.edu.cn (Fei Gao)

Introduction

In the natural world, higher organisms demonstrate remarkable intelligence in motion planning, enabling them to navigate coherently and naturally in various environments. Whether it is a dense forest or complex city streets, organisms effortlessly adapt to their surroundings, employing flexible and purposeful strategies to avoid obstacles and reach their destinations swiftly [1, 2, 3]. However, for current robots, when faced with complex scenes, their computational efficiency rapidly declines due to extensive interactions with obstacles, resulting in a temporary loss of motion direction or even getting stuck in place. This phenomenon evidently limits the application scope and effectiveness of robots in the real world.

In fact, when it comes to pathfinding in the environment, humans appear to be more direct and effortless compared to robots. Even for children, when provided with a map representing the environment, they can often naturally depict a curve on the map connecting the starting and ending points. This ability is closely related to humans' superior spatial comprehension [4, 5, 6].

Humans can effectively extract environmental features, identify key areas, and utilize intuition and experience to quickly find the path [7, 8]. This intelligent approach to path planning enables us to adapt flexibly to various situations. Whether in an open sports arena or within complex buildings, we can always quickly find the reasonable path and navigate accurately. In contrast, robots often construct a search tree by randomly sampling to gain an understanding of the connectivity within the environment. Subsequently, they employ graph search algorithms [9] to obtain a path.

Clearly, for humans, the idea of robots using sampling to analyze the environment and performing a search from the starting point is unimaginable and unnatural. Moreover, the efficiency of this planning method employed by robots is evidently negatively correlated with the complexity of the environment. As the environment becomes more complex, robots often require more sampling to obtain a more accurate description of the scene, which limits their performance in complex environments.

Certainly, in many cases, the paths humans directly acquire from their minds are not precise but rather serve as high-level global guidance. Therefore, during actual movement, humans often make fine adjustments to the previously obtained rough path to ensure smoothness and absolute safety. In other words, the overall navigation process in humans follows a hierarchical framework. For instance, when navigating through a building, individuals typically have a rough understanding of their intended direction in advance, such as knowing to make a right turn after passing through a specific corridor. This serves as high-level planning, providing overarching guidance for their actions. Subsequently, during the actual progression, humans continuously refine their movements to avoid non-structural obstacles.

Inspired by these observations, we propose a novel learning-enhanced hierarchical planning framework which enables efficient and stable trajectory generation with nonholonomic constraints in various complex scenarios. Within this framework, we define the front end as

path planning, which focuses on determining approximate topological routes as the high-level planner. The back end concentrates on trajectory optimization, refining the path into a high-quality, safe and time-parameterized trajectory dutifully considering the system's higher-order kinematic constraints. Similar to humans, we employed a neural network to model the path planning problem as an image generation task, effectively analyzing environmental features to depict paths. Our algorithm's key feature lies in its computational efficiency principle, decoupling it from the complexity of the environment, thus enabling stable and efficient generation of reasonable initial paths in arbitrary complex environments. Actually, these paths may not be entirely smooth or feasible, and the considered kinematic model is relatively coarse and low-dimensional. Therefore, subsequently, we designed a novel bi-layer trajectory optimization framework to further enhance the quality of the path. Our optimization approach effectively refines the path into a high-quality spatial-temporal optimal trajectory, considering more refined system higher-order kinematics. Moreover, our method combines differential flatness [10] for efficient problem solving and employs bi-layer smooth mappings to eliminate singularities in the flat model. This not only enhances numerical stability but also ensures the robust generation of feasible solutions.

Practically, the goal of an ideal motion planning system is to generate high-quality trajectories in a stable and robust manner, even in arbitrary complex environments. It is crucial to maintain low planning times, even as the environment becomes more complex, to ensure the stability of navigation systems. Robustness in solution quality is also essential, as we expect the planner to converge reliably to high-quality feasible solutions, regardless of the complexity of the planning problem. In the current state of the art, some motion planning methods also adopt a hierarchical structure [11, 12, 13, 14] similar to ours. However, both the front end and the back end face a trade off between stability, efficiency, and quality.

The traditional front end is typically modeled as a combinatorial optimization problem.

Such algorithms often solve this problem by searching or sampling within a low-dimensional configuration space [15, 16, 17, 18, 19, 20, 21, 22]. However, these methods may underperform in complex environments, resulting in an abundance of non-contributory samples and a significant increase in computational time and memory consumption. Moreover, to ensure the quality of the initial path and completeness in narrow environments, such algorithms often require a higher resolution in the state space, leading to combinatorial explosion, affecting time stability and reducing real-time performance. In contrast to these approaches that continuously explore and maintain a growth tree starting from the initial state, our method exhibits greater intelligence and directness. Inspired by the natural human ability to comprehend environments, we employ a neural network to simulate the process of humans drawing a curve connecting the starting and ending points. Our method leverages the macroscopic information from the environment and expert knowledge to efficiently depict reasonable paths in arbitrary complex environments. Undeniably, recent advancements [23, 24, 25, 26, 27, 28, 29, 30] demonstrate significant promise by leveraging neural networks to guide search or sampling processes. However, we argue that they do not fully exploit the potential of neural networks. These methods fundamentally still rely on sampling or search strategies, which impact efficiency and make the method inherently susceptible to the complexity of the environment, thereby reducing stability. In contrast, our algorithm directly leverages the network to depict paths from the environment without the need for cumbersome sampling or search procedures. Essentially, this decouples the problem complexity from the environment, ensuring higher efficiency and stability.

For the backend optimization, some methods depend excessively on prescriptive guidelines [31, 32, 33], circumscribing their adaptability to intricate and heterogenous scenarios. Owing to the complexities of nonholonomic kinematics, most of strategies mandate a markedly simplified motion paradigm [34, 35] or discrete motion process [36, 37, 38, 39, 40] to uphold algorithmic efficiency. Nonetheless, to ensure elevated executability and high success proba-

bilities in dense settings, these strategies necessitate refined discretization, thereby adversely negatively impact the temporal efficiency of the planning process. Recently, Han et al. [41] model fully differentiable trajectory optimization in the flat spaces with refined trajectory representations, validated for efficiency by extensive experiments. However, this method encounters singularities in its underlying principles, leading to numerical instability that hinders effective convergence to feasible solutions. In contrast, we design a novel dual-layer polynomial-based trajectory representation and remap the differential-flatness model smoothly. Our method 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 numerical stability and solution robustness.

In conclusion, we present an efficient and high-quality solution for motion planning of nonholonomic vehicles, which demonstrates a high level of time stability and robust convergence in arbitrary complex environments. In this paper, our research focuses primarily on the widely prevalent ackerman vehicles in autonomous driving. Furthermore, we demonstrate through experiments that our method is scalable to other vehicles with nonholonomic motion constraints, such as fixed-wing aircraft. Through extensive simulation and comparison, our deep path planning module shows a decoupling characteristic from the complexity of the environment. Compared to traditional methods, our method has an order of magnitude improvement in efficiency and time stability when the environment is more complex. Our trajectory optimization module is capable of reliably obtaining high-quality solutions that fully satisfy higher-order kinematic constraints, even in narrow and complex maneuvering scenarios. In contrast, the flatness-based algorithm, under the same computation time setting, often suffer from numerical instability, which can potentially lead to violations of kinematic constraints by an order of magnitude.

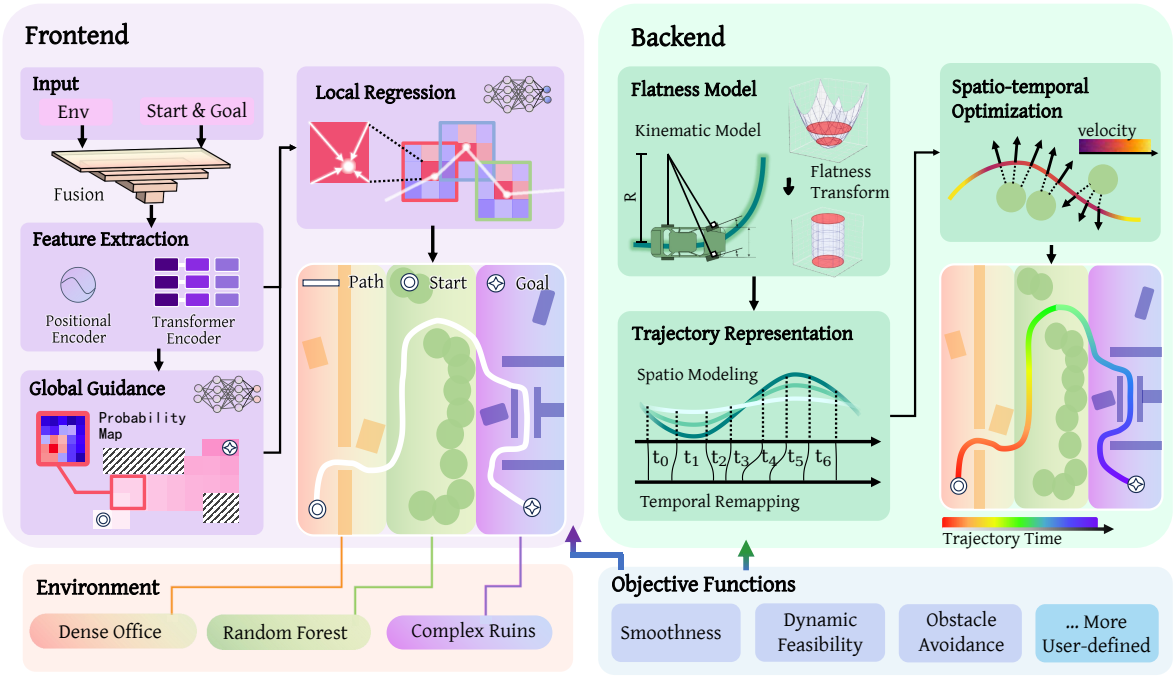

Fig. 1. Learning-enhanced hierarchical planning framework design. The frontend, acting as the high-level planner, focuses on determining an approximate topological route and is capable of generating initial paths efficiently and stably in complex environments such as dense office spaces, random forests, and complex ruins. The backend takes into consideration the higher-order kinematic constraints of the system, and incorporates a set of objective functions to optimize the path for space-time optimality, high quality, and safety.

Results

Experiment Details

We train our model on an Nvidia GTX 4090 GPU, while all simulation tests are conducted on a computer equipped with a GTX 2060 GPU, an Intel 10700 CPU, and running Ubuntu 20.04 operating system. For the comparative experiments in the subsequent path planning of Ackermann vehicles, our dataset contains three scenarios. In each scenario, we randomly generated 10000 environments. For each environment, we sampled 50 sets of different start and goal states to construct planning problems. For each problem, we utilize hybridAstar to generate an initial coarse curve and then employ the proposed trajectory optimization method to refine it, resulting in a high-quality solution. Subsequently, we uniformly sample 200 points along this trajectory, which serve as the ground truth for the supervised training of the frontend network. Regarding the backend optimization process, we employ an augmented Lagrangian approach within our dual-layer trajectory representation framework to relax the constraints of the original problem. Then, we efficiently solve the reformulated problem using the L-BFGS algorithm.

Time Stability in Complex Environments

We test our network in multiple scenarios, and the results show that even as the environment becomes more complex, our path planning algorithm maintains extremely low computation times and exhibits high stability. Furthermore, in complex scenarios, our algorithm exhibits a significant reduction in the variance of computation time compared to traditional baseline algorithms, highlighting its robustness across different planning problems.

The size of our testing environment is $20m * 20m$, with a resolution of $0.1m$. To verify the generalization capability and time stability of our network, we construct three different scenarios: (1) Random Forest: Approximately 60 irregular obstacles are present in the environment. (2) Dense Office: Randomly placed walls with narrow passages slightly larger than the robot's

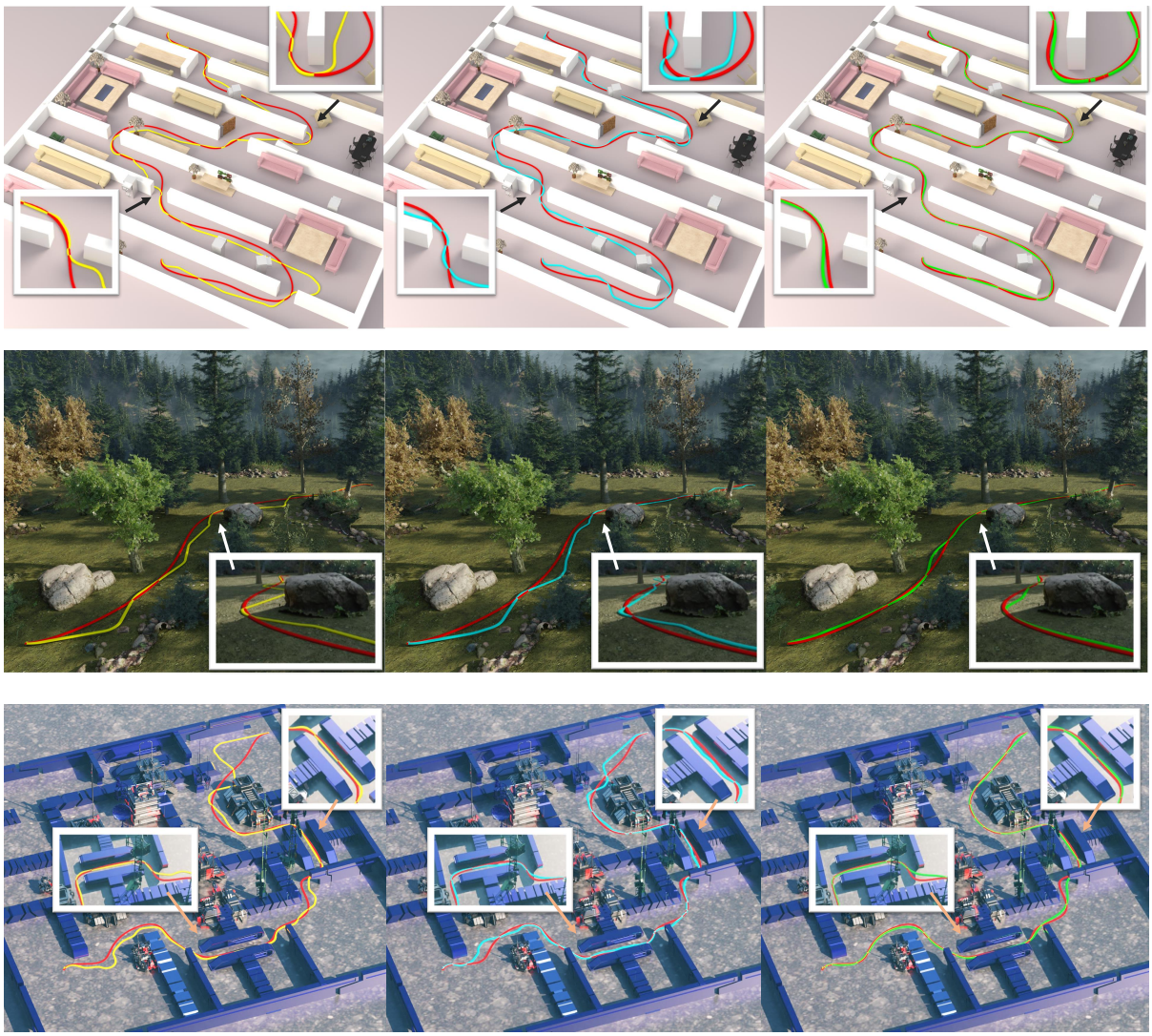

Fig. 2. The comparison of frontend path planning performance. From top to bottom, the scenes are as follows: dense office, random forest, and complex ruins. The red paths represent the ground truth, the yellow paths represent the planning results of Kino A*, the cyan paths represent the planning results of NN Kino A*, and the green paths represent the results generated by the proposed method.

size for traversal. (3) Complex Ruins: The robot often needs to take detours to reach the goal. We compare our model with a reliable hybrid A* [19] that has been extensively validated and has complete resolution coverage. Additionally, to demonstrate the advancement of our algorithm, we compare it with recent works that integrate neural networks and search techniques. Similar to our approach, this method use a transformer encoder to extract features of path planning problems. In all comparative experiments, the kinematic constraints are set equally for all algorithms, and the parameters are appropriately tuned for all approaches. Moreover, all paths generated by the frontend algorithms are further optimized using proposed trajectory optimization to enhance their quality. In each scenario, we test over 1000 instances, ensuring that the environments during testing are unseen during training. From Figure 1, it is evident that the paths output by our model are smoother and closer to the ground truth compared to other algorithms. This is because our algorithm eliminates the need for search and directly supervises the trajectory generation process using ground truth data, thereby imitating the behavior of ground true trajectories. Furthermore, we quantitatively analyze the computation time, variance, and success rate of various methods' modules. Since our final trajectory optimization algorithm's objective function consists of energy loss and execution time loss, the means of both losses are also measured to evaluate the quality of the final output trajectory. All these metrics are summarized in Table 1. From the table, it can be observed that our frontend algorithm maintains a high success rate while being capable of outputting paths with almost the same low computation time across different environments. In contrast, search-based algorithms, as a comparison, experience a significant increase in computation time as the environment becomes more complex due to their lack of holistic understanding of the environment and the consumption of numerous irrelevant samples. Although the second algorithm utilizes a network to extract key areas, it still relies on discrete state space search fundamentally. Therefore, in highly constrained environments, a large number of permutations and combinations are still required to produce solutions,

Table 1: **Quantitative benchmarks in various scenarios.**

Method	Front End					Back End				Success
	Seach Time (ms)	Model Time (ms)	Total Time (ms)	Time Ratio (%)	Variance in Time (ms^2)	Optimization Time (ms)	Energy Cost	Time Cost	Total Cost	Rate (%)
Random Forest										
Proposed	**0**	6.896	**6.896**	**100.0**	**1.733**	**17.58**	**38.18**	**218.0**	**256.18**	97.9
HbridA*	18.98	0	18.98	275.2	1105	20.58	41.10	234.0	275.1	97.7
THybridA*	12.58	**4.438**	17.02	246.8	1201	19.91	39.20	223.8	263.0	**98.6**
Complex Ruins										
Proposed	**0**	6.830	**6.830**	**100.0**	**1.389**	**24.00**	**53.37**	**302.3**	**355.67**	**98.0**
HbridA*	156.6	0	156.6	2293	16687	32.71	63.82	364.7	428.52	95.9
THybridA*	47.72	**4.478**	52.20	764.3	1822	30.62	65.24	367.5	432.74	97.9
Dense Office										
Proposed	**0**	7.193	**7.193**	**100.0**	**1.755**	**27.68**	**58.32**	**326.3**	**384.6**	**97.5**
HbridA*	277.1	0	277.1	3852	31232	39.92	67.64	418.3	485.9	79.6
THybridA*	90.60	**4.560**	95.16	1323	4417	35.78	62.42	391.2	453.62	93.4

lowering time stability. Moreover, due to the uncertainty in computation time of search algorithms, they exhibit significant time variance even for a single scenario. This instability makes it difficult to predict algorithm performance, consequently affecting the robustness of the entire navigation system. Furthermore, since search algorithms operate in a low-dimensional search space, their loss functions often differ from backend optimization, resulting in suboptimal paths assumed to be optimal by the frontend. In contrast, our algorithm directly mimics the behavior of backend optimization, reducing the gap between frontend and backend. The table also indicates that in all scenarios, our frontend algorithm requires the least amount of time for trajectory optimization. Moreover, the loss function of our final output trajectory is minimized, indicating the highest quality.

Application for Fixed Wing

We also verify the network's compatibility with other types of environmental descriptions, such as terrain elevation maps, and extend its application to large-scale fixed-wing navigation tasks. The results show that our algorithm can find a near-optimal solution in a $20km * 20km$ environ-

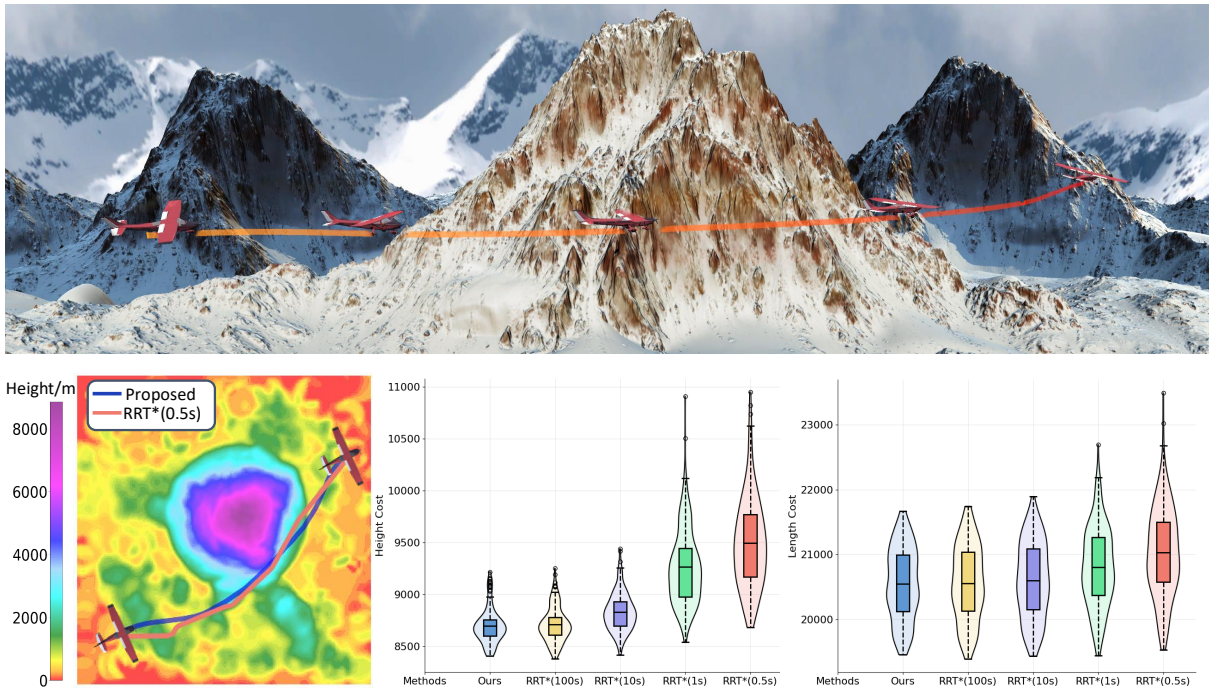

Fig. 3. Visualization of fixed wing applications. The top figure presents a visualization of fixed-wing flight in Unity3D, demonstrating the ability of fixed-wing aircraft to navigate around peaks to ensure safety. The figure in the bottom left corner displays a height map of the planning scene, where the heights of different locations are represented by colors. Additionally, we also depict the path generated by RRT* planning with a computation time of 0.5 seconds as an orange-red line for comparison. The figure in the bottom right corner showcases a comparative analysis of our method and RRT* at different computation times in terms of path length and height cost.

ment in just $50ms$, while traditional sampling algorithms require approximately $100s$ to achieve similar results, resulting in an efficiency improvement of around 2000 times.

In this study, we consider a common scenario where a fixed-wing aircraft is required to traverse mountainous and hilly terrains, ultimately reaching a target state at a given altitude, as shown in Fig. 3. To highlight our algorithm's superiority in this problem, we compare it with the classical sampling-based algorithm RRT* [17], which is theoretically proven to be optimal with infinite samples. Considering the minimum turning radius constraint of the fixed-wing aircraft, we modify the connection between two states in RRT* from a straight line segment to a Dubins

curve [42]. In this case, ensuring safety and communication with the ground, we aim for the fixed-wing flight path to be not only smooth and short but also to select lower altitude or flatter terrain whenever possible to achieve higher clearance from the ground. Therefore, we employ two metrics to measure the path: length cost, which quantifies the path length, and height cost, which measures the elevation of the ground. A smaller height cost indicates that the path traverses lower-altitude terrains, resulting in a higher relative clearance and increased safety. Since the RRT* algorithm exhibits asymptotic optimality, we allocate different computation time budgets ($0.5s, 1s, 10s, 100s$) for it. We conduct 100 experimental trials and visualize the length cost and the height cost relative to the planned path using violin plots, as shown in Fig. 3. From the perspective of the two loss functions, the quality of the path directly outputted by our model is comparable to that of RRT* with a computation time of $100s$. However, in this scenario, our model only requires $50ms$ of inference time, leading to approximately a 2000-fold improvement in efficiency compared to traditional algorithms

Numerical Stable Trajectory Optimization

Although we obtain a rough path quickly through the front end, backend trajectory optimization is still necessary to improve its quality and ensure compliance with finer kinematic models and collision avoidance. Some existing work [41] simplifies the trajectory optimization problem using differential flatness, effectively eliminating motion equation constraints and greatly increasing computational efficiency. However, this modeling approach has inherent singularities that prevent guaranteeing complete feasibility of the constraints. For instance, as shown in Fig. 4b, it relies on inferring the yaw angle from the direction of the velocity, which fails when the velocity is zero, leading to numerical instability during solution computation. In contrast, we introduce intermediate variables (pseudo velocity) to re-smoothly map the flat model and design a specialized trajectory representation. This approach ensures efficient optimization

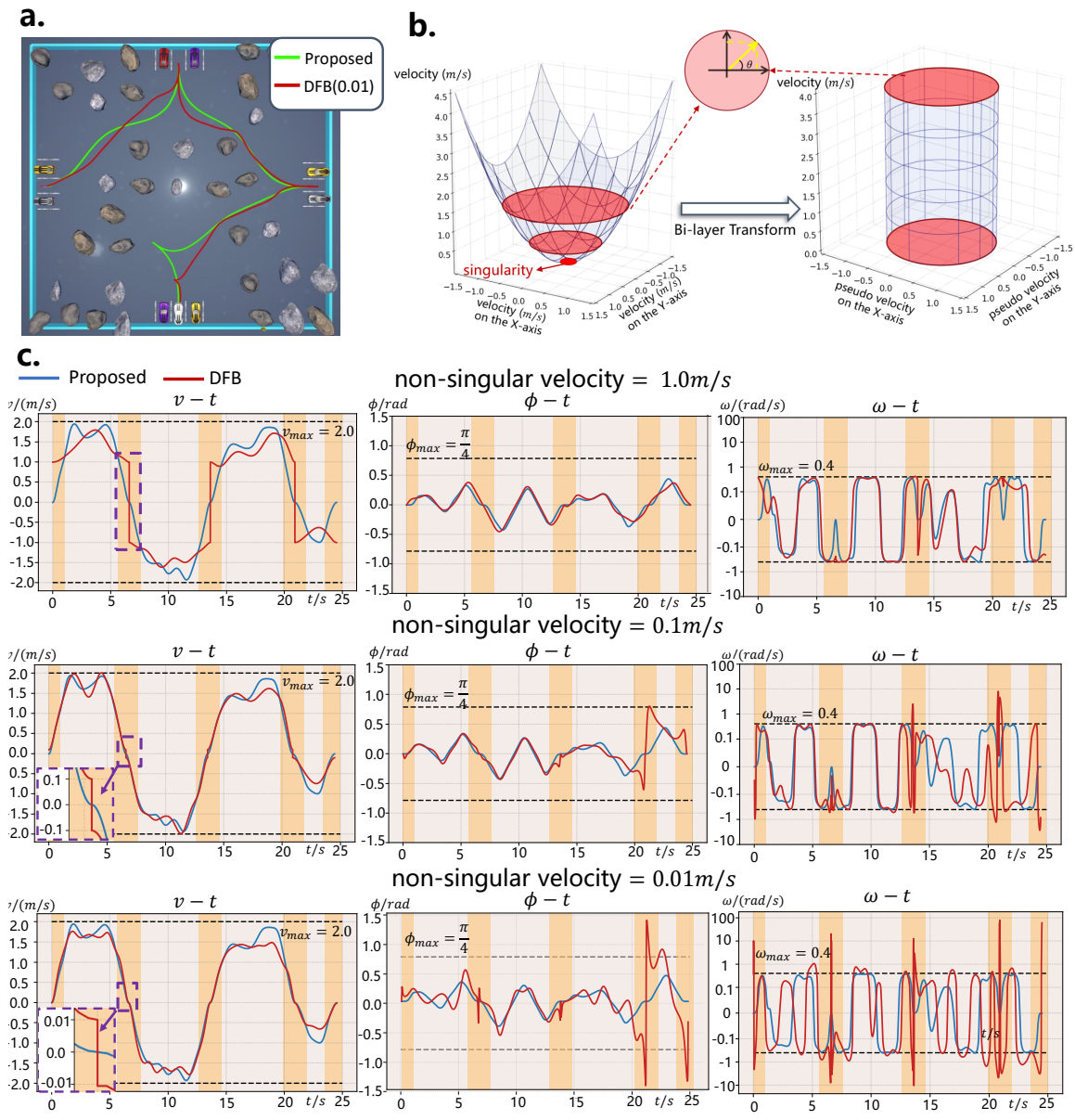

Fig. 4. Comparison visualization of our method and DFB. a. The visualization of planned trajectories. Here, for DFB, the switching speed during forward and backward motion is set to $0.01m/s$ to avoid computational singularities. **b.** Visualization of singular point elimination in the flat space. Prior to the transformation, when the vehicle's velocity is 0, singular points arise, preventing the calculation of the vehicle's heading angle. However, by employing our bi-layer transform and introducing pseudovelocity, the magnitude of the pseudovelocity always surpasses a threshold, allowing for the recovery of the vehicle's heading angle without encountering singularities, regardless of the actual velocity. **c.** A comparison of velocity, steer angle, and steer angle rate curves between our trajectory and the trajectories generated by DFB under different parameters.

while fundamentally resolving the problem, allowing stable generation of feasible solutions in highly complex environments. We rigorously validate our trajectory optimization algorithm in a challenging scenario that involves multiple forward and backward vehicle maneuvers to highlight its contributions. The results demonstrate that our algorithm achieves better numerical stability and robust generation of feasible solutions compared to a powerful baseline differential flat-based (DFB) trajectory optimization method [41] that has been extensively validated through experiments.

To replicate a common parking scenario observed in real-life, we construct a testing environment as illustrated in Fig. 4.a. In this environment, this vehicle is instructed to depart from a designated starting point and successively approach predetermined intermediate points with prescribed yaw angles. Finally, it is required to reverse into a specified endpoint. From this figure, it is evident that the presence of singular points leads to more convoluted trajectories generated by DFB planning. In contrast, our method produces significantly smoother trajectories. To ensure fairness, the optimization time for each method is fixed at $75ms$, and the total trajectory time and kinematic constraints are kept consistent across the methods. Since the baseline method [41] faces computational singularity when the velocity is zero, we fix it at a small value (non-singular velocity) during forward and backward motion and analyze the impact of this parameter.

We present curves depicting the variation of velocity v, steer angle ϕ, and steer angle rate ω of the planned trajectory over time for different cases, as shown in Fig. 4.c. These curves intuitively reveal the numerical instability caused by singularities. When non-singular velocity is $0.1m/s$, the steer angle rate of the trajectory can no longer be strictly satisfied. When non-singular velocity is $0.01m/s$, the steer angle constraint is also violated, and higher-order steer angle rates exceed the constraints by an order of magnitude. Although setting a larger non-singular velocity $= 1.0m/s$ can avoid singularity issues, it introduces discontinuity during

motion, resulting in severe mechanical component vibrations. In contrast, our method consistently exhibits good numerical stability, ensuring smooth velocity profiles and robust convergence to kinematically feasible solutions. In fact, kinematic infeasibility of the trajectory or abrupt changes in velocity can hinder the effective execution of the trajectory by lower-level controllers, leading to increased tracking errors and collision risks. To demonstrate this point, we introduce an model predictive control (MPC)-based controller to measure this error. According to the statistical data provided in our supplementary information, our method achieves a 90% reduction in maximum position tracking error and an 80% reduction in maximum angle tracking error compared to the baseline DFB, enhancing the reliability and completeness of the entire navigation system.

Discussion

Compared to humans who directly analyze spatial information and determine the approximate motion direction, robots always rely on extensive sampling or searching to understand the environment, which appears clumsy and counterintuitive. Therefore, drawing inspiration from the human thought process, we design a neural network to directly generate paths from the environment without the need for extensive sampling. We conduct numerous comparative experiments to validate our method's efficiency and stability advantages over traditional algorithms in various environments. Even as the environment becomes more complex, our path planning algorithm maintains almost the same computation time, while the baseline method's efficiency decreases by several orders of magnitude. Moreover, we conduct ablation experiments to analyze the impact of specific network structures on the results, and the results show that our special design effectively reduces the loss by 50%compared to the baseline. Further details on this ablation study can be found in the supplementary information.

For motion planning in robotics, the core objective is to obtain a high-quality trajectory that

satisfies a set of hard constraints, such as collision avoidance and kinematic feasibility. However, achieving this objective directly with a neural network is challenging. This is because, given a fixed model size, neural networks have inherent limitations in their learning capacity, making it difficult to converge to the optimal solution that satisfies all constraints. Therefore, we introduce backend optimization to further refine the path outputted by the frontend network into a fully constraint-compliant high-quality trajectory. Our backend trajectory optimization algorithm inherits the efficiency of flat-based algorithms while avoiding the numerical instability caused by singular points, enabling robust convergence to feasible solutions even in extreme scenarios.

Methods

Neural Path Planning

Our learning-based path planner takes as input the navigation start and goal state, as well as the grid-based environment , and directly outputs rough solutions. Moreover, each state point in the path is associated with the $\mathbb{SE}(2)$ state of the robot. Inspired by the work [30], we adopt a start-goal encoding strategy by highlighting patches on a tensor of size $H * W$. 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.

We introduce a self-attention mechanism to capture important relationships and patterns within input sequences. By stacking multiple layers and employing a multi-head attention mechanism, we encode environmental and odometry information at different abstraction levels, thereby better understanding the structure and semantics of the environment. The self-attention mechanism allows the model to globally capture information when processing sequence data, rather than being limited to a fixed environment size. This aids the model in better understanding the meaning of input information and optimizing space across different environments.

This mechanism enables parallel computation, performing the same reinforcement operations across all other positions within each optimization subdomain, greatly enhancing the model's computational efficiency. It is easy to embed this mechanism into other models and deploy it onto robots, providing deployment strategies for embodied robots. This mechanism reinforces learning efficiency and capability, especially advantageous when dealing with complex, high-dimensional, or even continuous state and action spaces.

Through the utilization of course learning, neural networks have acquired the capability to generate solutions even in adversarial circumstances. Consequently, we have redefined the problem of motion planning as a challenge in generative modeling. The incorporation of course learning has not only bolstered the network's generative capacity but also maintained consistency in training. As training progresses, the network's understanding of the environment deepens, enabling it to grasp the constraints within the optimization domain. Furthermore, as the workload of course learning gradually escalates, the network progressively enhances its ability to mitigate noise in noisy environments, thereby improving its overall performance.

Neural Network Architecture

Our network comprises three main components: the Feature Extraction Layer (FEL), the Global Distribution Layer (GDL), and the Local Correction Layer (LCL). 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 [43], our network follows a two-stage inference architecture. Initially, we uniformly partition environment into region proposals.

Subsequently, we employ GDL to obtain the probability distribution of each state point with respect to the region proposals. GDL is a generative functional network designed to generate feasible solutions within the optimization domain, with its output solutions conforming to

the total probability Bayesian formula. To enable GDL to find solutions within the optimization space, we incorporated real instances into the training process. The GDL network serves the purpose of optimizing and selecting feasible regions, thus accelerating the overall network training speed.

Next, leveraging LCL, we further obtain the accurate position of the state point. LCL is a network for local denoising. It refers to the results of global probability optimization and acts as a mapping function by limiting the local receptive field for denoising. Due to the self-attention of LCL being focused on local positions, this receptive field serves the purpose of local denoising. LCL is an efficient self-organized network, and its output contains parallel denoising results for multiple path points. To ensure the dynamical continuity of the network's forward propagation process, we integrated median filtering into the image generation process of the network.

Loss Functions

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. Therefore, our loss function $\mathcal{L}$ consists of supervised terms that mimic the ground truth and unsupervised terms that purely measure the validity of the path:

$$\begin{aligned}\mathcal{L} = w_{ce}\mathcal{L}_{ce} + w_{mse}\mathcal{L}_{mse} + w_{smo}\mathcal{L}_{smo} + w_{hol}\mathcal{L}_{hol} \\ + w_{cur}\mathcal{L}_{cur} + w_{uni}\mathcal{L}_{uni} + w_{obs}\mathcal{L}_{obs},\end{aligned} \tag{1}$$

where w_* is the weight corresponding to each loss. Here, $\mathcal{L}_{ce}$ is the anchor point classification loss. Because we model the role of the global distribution layer as a multi-classification problem, $\mathcal{L}_{ce}$ is used to maximize the probability of the region containing the ground truth point. $\mathcal{L}_{mse}$ is the path supervision loss which is utilized to supervise the mean squared error between the $\mathbb{SE}(2)$ states of the points on the output path and their corresponding ground truth values.

$\mathcal{L}_{smo}$ is the smoothness loss which is employed to enhance the smoothness of the planned path. $\mathcal{L}_{hol}$ is the nonholonomic dynamic loss to emphasize the alignment between the direction of the line connecting adjacent points and the orientation angle direction. $\mathcal{L}_{cur}$ is the curvature constraint loss to to penalize exceeding the specified curvature threshold. $\mathcal{L}_{uni}$ is the uniform loss, which is employed to encourage a more uniform spatial distribution of points along the path. This is achieved by penalizing the variance of the points' positions in space. $\mathcal{L}_{obs}$ is the obstacle avoidance loss, which is utilized to penalize potential collisions with obstacles. The specific rigorous mathematical expressions and corresponding weights for each loss can be found in the supplementary information.

Bi-Layer Trajectory Optimization

Remapping of differential flat models

The characteristic of differentially flat systems is the ability to analytically represent the system state using the combination of flat outputs and their finite-dimensional derivatives. Here, we take the example of Ackermann kinematics [44]. By selecting the flat outputs $\boldsymbol{\sigma} := (p_x, p_y)^{\mathrm{T}}$ as the position at the center of the real wheels, , the other high-order states of the robot during forward motion can be expressed as follows:

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

$$a = \frac{\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}} \dot{\boldsymbol{\sigma}}_{|t}}{||\dot{\boldsymbol{\sigma}}_{|t}||_2}, \phi = \arctan\left(\frac{\ddot{\boldsymbol{\sigma}}_{|t}^{\mathrm{T}} \mathbf{B} \dot{\boldsymbol{\sigma}}_{|t} L}{||\dot{\boldsymbol{\sigma}}_{|t}||_2^3}\right), \tag{2b}$$

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

$\dot{\boldsymbol{\sigma}}_{|t}, \ddot{\boldsymbol{\sigma}}_{|t}$ and $\dddot{\boldsymbol{\sigma}}_{|t}$ are the first, second, and third derivatives of the flat output w.r.t time, respectively. $\mathbf{B} := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ is an auxiliary antisymmetric matrix for computational convenience. Besides, 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.

While leveraging differential flatness can accelerate the optimization process, it should be noted that when the velocity approaches zero, certain states such as Eq. (2b) and Eq. (2c) may encounter singularities. By contrast, we introduce a *Pseudo Arc* to indirectly deriving the flat model which fundamentally eradicates the singularity issues:

$$\boldsymbol{\sigma} = \boldsymbol{\gamma}(s), s = s(t). \tag{3}$$

Here, $s \in \mathbb{R}^+$ is the *Pseudo Arc*. Then the finite-dimensional derivative of the flat output can also be derived as follows:

$$\dot{\boldsymbol{\sigma}}_{|t} = \dot{\boldsymbol{\gamma}}_{|s}\dot{s}_{|t}, \tag{4a}$$

$$\ddot{\boldsymbol{\sigma}}_{|t} = \dot{\boldsymbol{\gamma}}_{|s}\ddot{s}_{|t} + \dot{s}_{|t}^2\ddot{\boldsymbol{\gamma}}_{|s}, \tag{4b}$$

$$\dddot{\boldsymbol{\sigma}}_{|t} = \dddot{s}_{|t}\dot{\boldsymbol{\gamma}}_{|s} + 3\dot{s}_{|t}\ddot{\boldsymbol{\gamma}}_{|s}\ddot{s}_{|t} + \dot{s}_{|t}^3\dddot{\boldsymbol{\gamma}}_{|s}, \tag{4c}$$

$\dot{\boldsymbol{\gamma}}_{|s}, \ddot{\boldsymbol{\gamma}}_{|s}$ 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. (4a-4c) into Eq. (2a-2c), thus modifying the differential flatness model. Taking ϕ and ω as examples, they are redefined as follows:

$$\phi = \arctan\left(\frac{\ddot{\boldsymbol{\gamma}}_{|s}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}L}{||\dot{\boldsymbol{\gamma}}_{|s}||_2^3}\right), \tag{5a}$$

$$\omega = L\frac{\dddot{\boldsymbol{\gamma}}_{|s}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}||\dot{\boldsymbol{\gamma}}_{|s}||_2^3 - 3\ddot{\boldsymbol{\gamma}}_{|s}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}\ddot{\boldsymbol{\gamma}}_{|s}^{\mathrm{T}}\dot{\boldsymbol{\gamma}}_{|s}||\dot{\boldsymbol{\gamma}}_{|s}||_2}{||\dot{\boldsymbol{\gamma}}_{|s}||_2^6 + (\ddot{\boldsymbol{\gamma}}_{|s}^{\mathrm{T}}\mathbf{B}\dot{\boldsymbol{\gamma}}_{|s}L)^2}\dot{s}_{|t}. \tag{5b}$$

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. Consequently, the denominator of Eq. (5a-5b) 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 bi-layer piece-wise polynomials.

Bi-Layer Polynomial-Based Trajectory Representation

To ensure sufficient degrees of freedom and smoothness, we parameterize $\boldsymbol{\gamma}(s)$ and $s(t)$ using piecewise polynomials, thereby establishing a compact nonlinear optimization problem. We formulate the trajectory $\boldsymbol{\gamma}(s)$ as a M-piece polynomial with degree $D = 2u - 1$, which is parameterized by the *Pseudo Arc* $\delta \boldsymbol{s} = (\delta s_1, ..., \delta s_M)^{\mathrm{T}} \in \mathbb{R}^M$ corresponding to each piece and the coefficient matrix $\boldsymbol{c}^p = \left((\boldsymbol{c}_1^p)^{\mathrm{T}}, ..., (\boldsymbol{c}_M^p)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2Mu \times 2}$. Similarly, the *Pseudo Arc* $s(t)$ is represented as a one-dimensional and time-uniform M-piece polynomial, parameterized by the time interval for each piece δT and coefficient matrix $\boldsymbol{c}^s = \left((\boldsymbol{c}_1^s)^{\mathrm{T}}, ..., (\boldsymbol{c}_M^s)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2Mu}$. Moreover, we consider a strict correspondence between each piece of the bi-layer piece-wise polynomials, such that for every time interval δT, the robot should traverse the corresponding *Pseudo Arc*:

$$s(i * \delta T) = \sum_{j=1}^{i} \delta s_j, \forall i \in \{1, 2, 3, ..., M\}, \tag{6}$$

Based on the above modeling, the i-th piece of $\boldsymbol{\gamma}_i$ is represented as:

$$\begin{aligned} \boldsymbol{\gamma}_i(s_i) &:= (\boldsymbol{c}_i^p)^{\mathrm{T}} \boldsymbol{\beta}(s_i - \sum_{j=1}^{i-1} \delta s_j), \\ s_i(t) &:= (\boldsymbol{c}_i^s)^{\mathrm{T}} \boldsymbol{\beta}(t), \forall t \in [0, \delta T], \end{aligned} \tag{7}$$

where $\boldsymbol{\beta}(x) := \left(1, x, x^2, ..., x^N\right)^{\mathrm{T}}$ is a natural basis function. Moreover, due to the strict correspondence between each piece of the bi-layer polynomials, we can further derive the following equation from Eq. (6):

$$s_i(0) := \sum_{j=1}^{i-1} \delta s_j, s_i(\delta T) := s_i(0) + \delta s_i \tag{8}$$

where $s_1(0)$ is set as 0. Furturemore, the M-piece polynomial $\boldsymbol{\gamma}$ is obtained:

$$\begin{aligned}\boldsymbol{\sigma}(t) = \boldsymbol{\gamma}(s(t)) = \boldsymbol{\gamma}_i(s(t)), s(t) = s_i(t - (i-1) * \delta T),\\ t \in [(i-1) * \delta T, i * \delta T).\end{aligned} \tag{9}$$

With motion feasibility constraints, the minimum control effort problem based on the modified flatness model Eq. (5a-5b), incorporating first-order temporal regularization, is formulated as a nonlinear constrained optimization:

$$\min_{\mathbf{c}^p, \mathbf{c}^s, \delta\mathbf{s}, \delta T \in \mathbb{R}^+} J = \int_0^{T_s} \boldsymbol{\sigma}_{|t}^{(u)}(t)^{\mathrm{T}} \boldsymbol{\sigma}_{|t}^{(u)}(t) dt + \rho T_s \tag{10a}$$

s.t.

$$\mathcal{B}(\boldsymbol{\sigma}_{|t}(0) ... \boldsymbol{\sigma}_{|t}^{(u-1)}(0), \boldsymbol{\sigma}_{|t}(T_s) ... \boldsymbol{\sigma}_{|t}^{(u-1)}(T_s)) = 0, \tag{10b}$$

$$\mathcal{T}(\boldsymbol{\gamma}_1 ... \boldsymbol{\gamma}_M, s_1 ..., s_M, \delta\mathbf{s}, \delta T) = 0, \tag{10c}$$

$$||\dot{\boldsymbol{\gamma}}_s(s(t))||_2 > \alpha, \tag{10d}$$

$$\dot{s}_{|t}(t) \geq 0, \tag{10e}$$

$$\mathcal{G}(\boldsymbol{\gamma}(s(t)), ..., \boldsymbol{\gamma}_{|s}^{(u)}(s(t)), s(t), ..., s_{|t}^{(u)}(t)) \preceq 0, \tag{10f}$$

$u = 3$ 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. (10b) is the boundary condition representing the initial and final state constraints of the trajectory. Eq. (10c) denotes the continuity constraints at the junctions of the piece-wise polynomials. Eq. (10d) is a minimum *Pseudo Velocity* constraint introduced to avoid singularities, where α is a threshold value. Eq. (10e) correspond to positive-definite constraints on *Pseudo Velocity*. $\mathcal{G}$ encompasses common inequality constraints considered in trajectory planning problems, including dynamic feasibility and obstacle avoidance constraints.

References

[1] E. A. Capaldi, G. E. Robinson, S. E. Fahrbach, Neuroethology of spatial learning: the birds and the bees, *Annual review of psychology* **50**, 651–682 (1999).

[2] J. Wiener, S. Shettleworth, V. P. Bingman, K. Cheng, S. Healy, L. F. Jacobs, K. J. Jeffery, H. A. Mallot, R. Menzel, N. S. Newcombe, *Animal thinking: Contemporary issues in comparative cognition* (The MIT Press, 2011), pp. 51–76.

[3] K. M. Chu, S. H. Seto, I. N. Beloozerova, V. Marlinski, Strategies for obstacle avoidance during walking in the cat, *Journal of Neurophysiology* **118**, 817–831 (2017).

[4] M. I. Sereno, R.-S. Huang, A human parietal face area contains aligned head-centered visual and tactile maps, *Nature neuroscience* **9**, 1337–1343 (2006).

[5] B. A. Wandell, S. O. Dumoulin, A. A. Brewer, Visual field maps in human cortex, *Neuron* **56**, 366–383 (2007).

[6] C. D. Harvey, F. Collman, D. A. Dombeck, D. W. Tank, Intracellular dynamics of hippocampal place cells during virtual navigation, *Nature* **461**, 941–946 (2009).

[7] A. D. Ekstrom, S. Y. Bookheimer, Spatial and temporal episodic memory retrieval recruit dissociable functional networks in the human brain, *Learning & memory* **14**, 645–654 (2007).

[8] R. A. Epstein, C. I. Baker, Scene perception in the human brain, *Annual review of vision science* **5**, 373–397 (2019).

[9] S. K. Debnath, R. Omar, N. B. A. Latip, S. Shelyna, E. Nadira, C. Melor, T. K. Chakraborty, E. Natarajan, A review on graph search algorithms for optimal energy ef-

ficient path planning for an unmanned air vehicle, *Indonesian Journal of Electrical Engineering and Computer Science* **15**, 743–749 (2019).

[10] M. van Nieuwstadt, M. Rathinam, R. M. Murray, Differential flatness and absolute equivalence, *Proceedings of 1994 33rd IEEE Conference on Decision and Control* (IEEE, 1994), vol. 1, pp. 326–332.

[11] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, X. Peng, Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework, *IEEE Transactions on Intelligent Transportation Systems* **23**, 11970-11981 (2022).

[12] R. He, J. Zhou, S. Jiang, Y. Wang, J. Tao, S. Song, J. Hu, J. Miao, Q. Luo, TDR-OBCA: A reliable planner for autonomous driving in free-space environment, *2021 American Control Conference (ACC)* (IEEE, 2021), pp. 2927–2934.

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

[14] S. Zhang, Z. Jian, X. Deng, S. Chen, Z. Nan, N. Zheng, Hierarchical motion planning for autonomous driving in large-scale complex scenarios, *IEEE Transactions on Intelligent Transportation Systems* **23**, 13291–13305 (2021).

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

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

[17] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, *The international journal of robotics research* **30**, 846–894 (2011).

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

[19] D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, Path planning for autonomous vehicles in unknown semi-structured environments, *The International Journal of Robotics Research* **29**, 485–501 (2010).

[20] Z. Ren, S. Rathinam, M. Likhachev, H. Choset, Multi-objective safe-interval path planning with dynamic obstacles, *IEEE Robotics and Automation Letters* **7**, 8154–8161 (2022).

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

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

[23] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, A. Kanezaki, Path planning using neural a* search, *International conference on machine learning* (PMLR, 2021), pp. 12029–12039.

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

[25] J. Wang, X. Jia, T. Zhang, N. Ma, M. Q.-H. Meng, Deep neural network enhanced sampling-based path planning in 3d space, *IEEE Transactions on Automation Science and Engineering* **19**, 3434–3443 (2021).

[26] D. Kim, K. Huh, Neural motion planning for autonomous parking, *International Journal of Control, Automation and Systems* **21**, 1309–1318 (2023).

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

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

[29] L. Li, Y. Miao, A. H. Qureshi, M. C. Yip, Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints, *IEEE Robotics and Automation Letters* **6**, 4496–4503 (2021).

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

[31] F. Gómez-Bravo, F. Cuesta, A. Ollero, A. Viguria, Continuous curvature path generation based on β-spline curves for parking manoeuvres, *Robotics and autonomous systems* **56**, 360–372 (2008).

[32] A. Gupta, R. Divekar, Autonomous parallel parking methodology for ackerman configured vehicles, *ACEEE International Journal on Communication* **1**, 1–6 (2010).

[33] C. Sungwoo, C. Boussard, B. d'Andréa Novel, Easy path planning and robust control for automatic parallel parking, *IFAC Proceedings Volumes* **44**, 656–661 (2011).

[34] W. Lim, S. Lee, M. Sunwoo, K. Jo, Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios, *IEEE Transactions on Intelligent Transportation Systems* **22**, 341–355 (2019).

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

[36] 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* **86**, 11–20 (2015).

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

[38] S. Shi, Y. Xiong, J. Chen, C. Xiong, A bilevel optimal motion planning (bomp) model with application to autonomous parking, *International Journal of Intelligent Robotics and Applications* **3**, 370–382 (2019).

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

[40] C. Sun, Q. Li, B. Li, 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* **23**, 3724–3736 (2021).

[41] Z. Han, Y. Wu, T. Li, L. Zhang, L. Pei, L. Xu, C. Li, C. Ma, C. Xu, S. Shen, F. Gao, An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments, *IEEE Transactions on Intelligent Transportation Systems* pp. 1–18 (2023).

[42] I. Lugo-Cárdenas, G. Flores, S. Salazar, R. Lozano, Dubins path generation for a fixed wing uav, *2014 International conference on unmanned aircraft systems (ICUAS)* (IEEE, 2014), pp. 339–346.

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

[44] M. Tanelli, M. Corno, S. Saveresi, *Modelling, simulation and control of two-wheeled vehicles* (John Wiley & Sons, 2014).