AUTONOMOUS VEHICLES

Hierarchically depicting vehicle trajectory with stability in complex environments

Zhichao Han[1,2]†, Mengze Tian[1]†, Zaitian Gongye[1], Donglai Xue[2], Jiaxi Xing[2], Qianhao Wang[1,2], Yuman Gao[1,2], Jingping Wang[1,2], Chao Xu[1,2], Fei Gao[1,2]*

The rapid development of autonomous robots has resulted in marked societal and economic benefits. However, enabling robots to navigate complex environments with human-like agility remains a formidable challenge. Unlike robots, humans excel at pathfinding because of their superior spatial awareness and their ability to leverage experience. Inspired by these observations, we designed a neural network to simulate the intuitive pathfinding abilities of humans, integrating global environmental information and previous experiences to identify feasible pathways. Experiments demonstrated that, unlike traditional algorithms whose efficiency deteriorates in complex settings, the proposed method maintains stable computational performance. To further enhance motion quality, we introduce a numerically stable spatiotemporal trajectory optimizer with a unique bilayer polynomial trajectory representation in flat space. This optimization leverages differential flatness to enhance efficiency and fundamentally eliminates singularities in the original problem, thereby robustly converging to continuous and feasible motion even in complex maneuvering scenarios. Our hierarchical motion planner, validated through large-scale maze experiments, combines front-end path planning with back-end trajectory refinement, achieving robust and efficient navigation. We anticipate that our planner will advance stable navigation for robots in complex environments, thereby propelling the progress of robotic autonomy.

INTRODUCTION

In the ever-evolving dance of nature, intelligent creatures exhibit an unmatched elegance and efficiency in motion planning. Their ability to navigate through dense forests or bustling city streets with ease and precision exemplifies their exceptional agility and adaptability (*1*–*3*). This capability contrasts with the challenges faced by modern robots, which often falter in complex environments. When confronted with convoluted obstacles, robots can become disoriented or immobilized, revealing a lack of practical deployment (*4*). For instance, a delivery robot in a crowded urban area may freeze or completely fail to accomplish its task, thus undermining its reliability and utility.

Intuitively depicting paths from environments

The disparity between human and robotic navigation is due to the superior spatial awareness of humans (*5*–*7*). Given a map, even young children can intuitively depict a route from start to end. Humans excel at extracting key environmental features by leveraging intuition and experience to identify viable paths quickly (*8*). This approach to path planning allows seamless adaptation to various settings, such as navigating the layout of a complex building or an open sports arena. In contrast, robots adhere to predefined rules to collect samples, strive to comprehend environmental connectivity, and subsequently find a path (*9*). This process is laborious and unnatural for humans. As the geometric complexity of the environment increases, robots must collect more samples to understand the scene, which reduces computational efficiency. Thus, although robots painstakingly piece together their surroundings, humans effortlessly perceive and navigate, underscoring a profound disparity in motion planning capabilities attributed to our advanced spatial cognition and utilization of experiential knowledge.

Inspired by these observations, we addressed the path planning problem by using a neural network to mimic the process of drawing a curve between the start and end points used by humans. This approach leverages macroscopic environmental information and expert knowledge to efficiently depict reasonable paths, regardless of the geometric complexity of environments. A key advantage of our method is temporal stability, which notably enhances the predictability and reliability of the navigation system.

Traditional path planning is often formulated as a combinatorial optimization problem, where algorithms typically construct a search tree from the starting point by iteratively exploring and sampling the low-dimensional configuration space (*10*–*17*). These methods frequently underperform in complex environments, resulting in excessive noncontributory sampling and increased computational and memory demands. Moreover, achieving high-quality solutions and completeness in narrow environments often requires a higher resolution of the state space, leading to a combinatorial explosion, degraded temporal stability, and limited real-time performance. Advances (*18*–*25*) have shown that neural networks can effectively guide the search or sampling process. However, these approaches do not fully exploit the capabilities of neural networks, because they fundamentally rely on traditional search or sampling, which makes their performance inherently susceptible to environmental complexity, thus reducing temporal stability. In contrast, our model directly extracts an initial guiding path from the environment, eliminating sampling and search and achieving near-constant inference time. By unifying path planning in the image domain, we normalize map scales and enhance geometric understanding for more effective obstacle avoidance. Recently, some studies have used generative models (*26*) to learn the spatial distribution of paths, which have been applied in embodied intelligence and task planning (*27*, *28*). However,

[1]Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University, Hangzhou, China. [2]Huzhou Institute of Zhejiang University, Huzhou, China.
*Corresponding author. Email: fgaoaa@zju.edu.cn
†These authors contributed equally to this work.

the iterative neural network inference required by these methods reduces real-time efficiency (*29*, *30*) and thus limits rapid response and replanning capabilities on computationally constrained onboard platforms (*31*, *32*).

Precisely refining intuitive paths into optimal trajectories

In real-life scenarios, even for humans, the paths intuitively conceived in the mind are neither precisely described nor kinematically feasible but instead serve as high-level guidance. During actual movement, humans frequently fine-tune these preliminary routes to ensure smooth and safe navigation. Consistent with previous studies (*33*, *34*), we hypothesize that human navigation operates within a hierarchical framework where high-level planning is followed by low-level corrective adjustments. For instance, rally drivers maintain a general understanding of the intended direction, such as anticipating a right turn after a straight stretch, which serves as high-level guidance. They subsequently fine-tune their driving to optimize both speed and safety.

Similarly, our neural network functions as a high-level path planner (front end), focusing on determining approximate topological routes. The paths generated by the front end may not be completely smooth or feasible, and the kinematic model considered is relatively coarse. To enhance this, a powerful trajectory optimization module (back end) is required to refine the path generated by the front end into a spatiotemporal optimal trajectory, meticulously considering the higher-order kinematic constraints of the system. However, existing trajectory optimization methods for nonholonomic vehicles face trade-offs between efficiency, generality, and convergence stability. Owing to the complexities of nonholonomic kinematics, most strategies require a markedly simplified motion paradigm (*35*, *36*) or a discrete motion process (*37*–*41*) to maintain algorithmic efficiency. Nonetheless, to ensure elevated executability and high success probabilities in dense settings, these strategies necessitate refined discretization, thereby negatively affecting the temporal efficiency of the planning process. In a previous study (*42*), we used minimum energy trajectory representations (*43*) to model the original trajectory optimization problem in flat spaces, which demonstrates a substantial efficiency advantage over conventional discrete motion methods through extensive experiments. However, this method encounters singularities in its underlying principles, leading to numerical instability that hinders effective convergence to feasible solutions in scenarios involving complex maneuvers. Drawing inspiration from this study, we used differential flatness for efficient problem-solving. However, unlike the previous study, we introduced auxiliary scale variables and specifically designed a distinctive bilevel piecewise polynomial-based trajectory representation to parameterize the robot's motion with respect to scale and the scale variable with respect to time. The innovation ensures higher-order continuity of motion while elegantly resolving the singularity issues inherent in the original optimization problem from a mathematical standpoint, thereby substantially enhancing the numerical stability of the optimization process. This enhancement enables our spatiotemporal optimization to consistently find feasible and high-quality trajectories in complex scenarios, whereas the baseline exhibits numerical instability and severe constraint violations.

Overall, as illustrated in Fig. 1, we integrated the proposed front end and back end to develop a learning-enhanced hierarchical motion planning system for nonholonomic vehicles, achieving stable computation time and optimization convergence in complex environments. This study primarily focuses on nonholonomic ground robots, such as Ackermann vehicles, but can be extended to other platforms with nonholonomic constraints, including fixed-wing aircraft, demonstrating strong adaptability across diverse environments and robotic systems. Extensive simulations and comparisons show that our deep path planning module decouples computational efficiency from environmental complexity. Our trajectory optimizer overcomes flat model singularities, stably converging to spatiotemporal optima that satisfy higher-order kinematic constraints, whereas baseline algorithms often exhibit numerical instability and violate these constraints.

RESULTS

Experimental overview and setup

To validate the efficiency and temporal stability of our front-end method, we conducted experiments using an Ackermann-steer vehicle in various scenarios, compared with other path planning methods. The results showed that as the complexity of the scenarios increased, our algorithm exhibited a substantial reduction in computation time compared with those of traditional baseline algorithms, highlighting its robustness across various planning problems. Furthermore, we refined the paths generated by each front-end method using our back-end trajectory optimization. The results showed that, compared with baseline algorithms, the initial paths generated by our method require considerably less optimization time to achieve top-quality trajectories. This indicates that our paths, when used as initial solutions, are of higher quality and more conducive to optimization. In addition, we extended the path-planning method to fixed-wing aircraft, demonstrating that it could directly plan feasible and safe paths on the basis of environmental elevation maps. In terms of balancing quality and efficiency, the traditional rapidly exploring random tree star (RRT*) algorithm took more time to achieve than the proposed method. Subsequently, we evaluated the numerical stability of the proposed back-end optimization method. We compared our trajectory optimization approach with recent flatness-based methods and constructed a complex scenario that required frequent forward and backward vehicle maneuvers. The results revealed that, owing to the presence of singularities, the trajectories planned by the flatness-based method inevitably exhibited abrupt changes in velocity. Additionally, these singularities led to numerical instability during the optimization process, resulting in distorted trajectories that did not completely satisfy dynamic constraints. However, the proposed method fundamentally addresses the singularity issues of the flatness model, enabling us to plan safe and completely smooth dynamically feasible trajectories even in complex scenarios. Finally, to verify the practicality of our planner, we deployed it in a large-scale outdoor maze scenario. The results demonstrated that the robot navigated efficiently and safely, showing its robustness to previously unseen obstacles and mapping errors. In terms of implementation, we trained our network on an NVIDIA RTX 4090 GPU, whereas all simulation tests were conducted on a computer equipped with an RTX 2060 GPU, an Intel 10700 CPU, and an Ubuntu 20.04 operating system.

Time stability in complex environments

In this benchmark, the size of the test environment was 20 m by 20 m, with a resolution of 0.1 m. To verify the generalization capability, we constructed three scenarios shown in Fig. 2A, as follows: (i) random forest: approximately 60 irregular obstacles were present in the

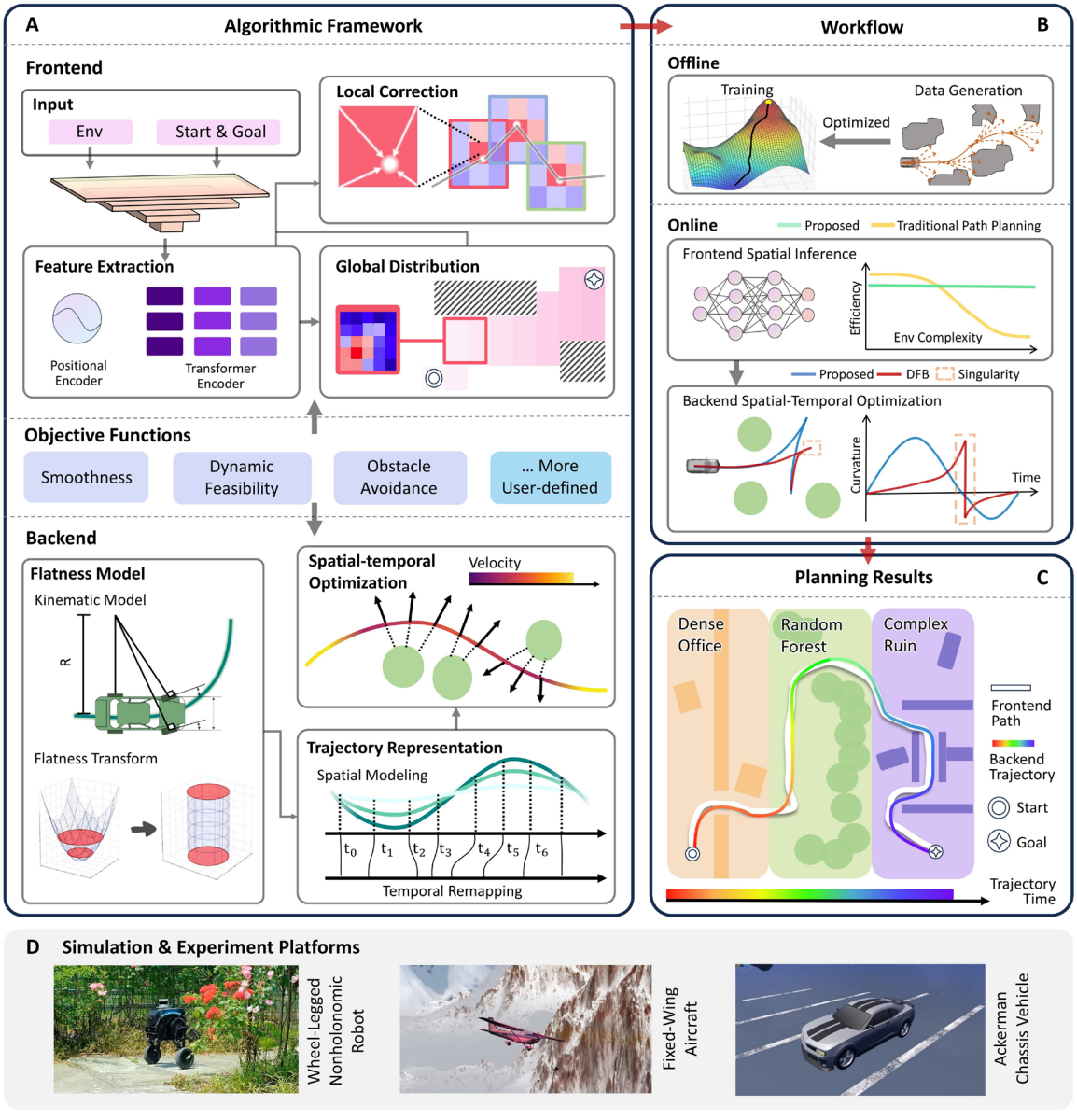

Fig. 1. System architecture and platform specifics. (**A**) Algorithmic framework: The front end includes feature extraction, global distribution, and local refinement layers. The back end is a spatiotemporal trajectory optimizer based on differential flatness, eliminating singularities. (**B**) Workflow: Initial paths are optimized as ground truth. Online inference generates a coarse curve, which is then optimized in space and time, avoiding singularities. (**C**) Planning results: The hierarchical planner is evaluated in dense office, random forest, and complex ruin environments. (**D**) Platforms: The planner adapts to various nonholonomic platforms.

environment; (ii) dense office: the environment contained randomly placed walls with narrow passages slightly larger than the size of the robot; (iii) complex ruins: the robot often needed to take detours to reach its goal. In each scenario, we randomly generated 10,000 environments, and for each environment, 50 sets of start and goal states were sampled to construct planning problems. For each problem, we discretized the configuration space with high precision and searched for an initial coarse curve (*44*, *45*), which was further refined using our trajectory optimization method. Subsequently, we uniformly sampled 200 points along each trajectory to serve as the ground truth for the supervised training of the front-end network. Additionally, we ensured that the environments for the comparative experiments were unseen during training.

We compared the proposed method with the classical Hybrid A* (*11*) and the recent THybrid A* (*19*), which combines neural networks with search. Similar to the proposed approach, THybrid A* uses a transformer (*46*) encoder to extract features from path planning problems. Additionally, all paths generated by the front-end algorithms were further optimized using the proposed trajectory optimization to enhance their quality. In each scenario, we tested 1000 previously unseen cases and depicted the paths generated by the front-end algorithms, as shown in Fig. 2A. The output of the

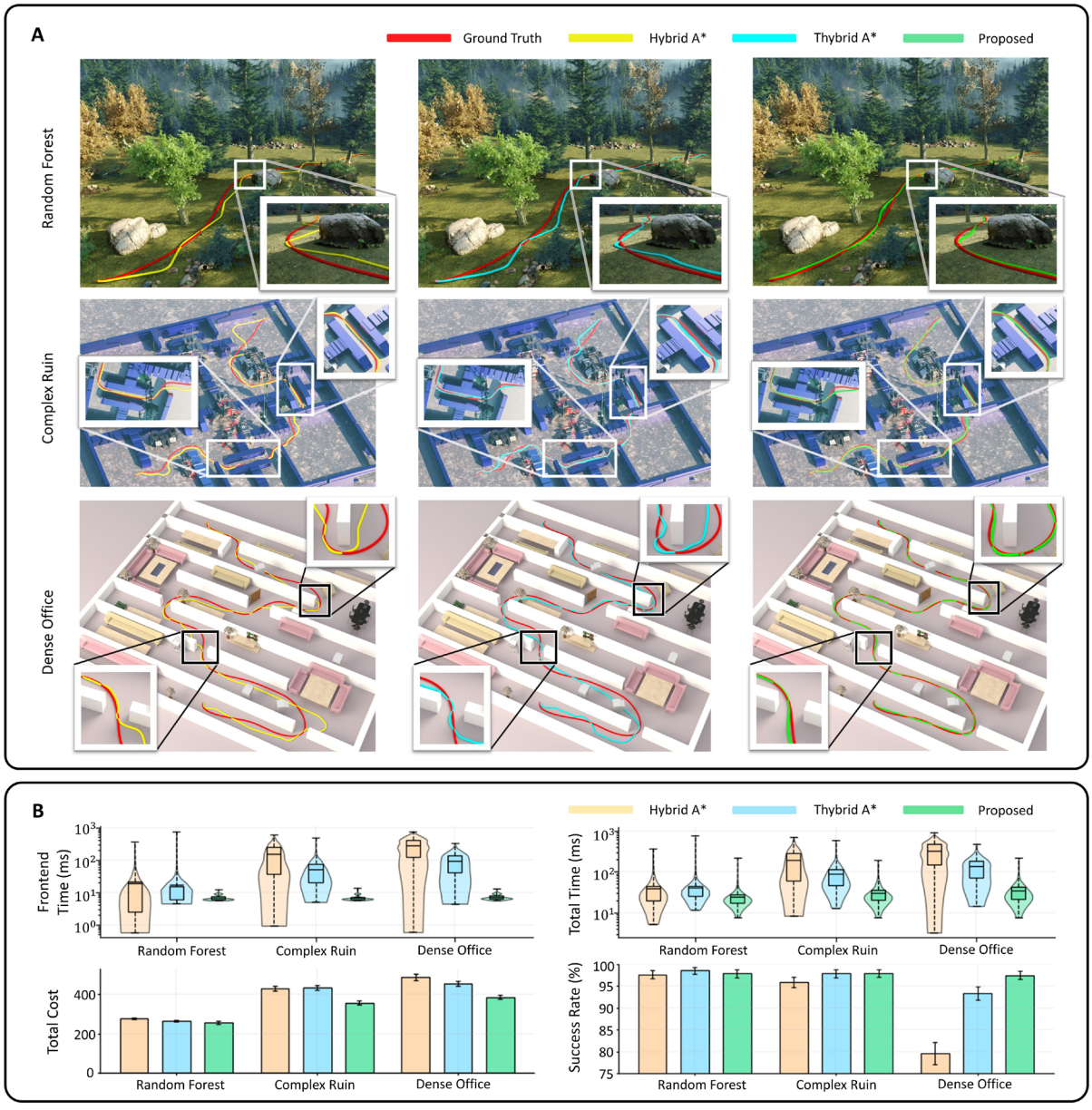

Fig. 2. Temporal stability in various complex scenarios. (**A**) Experimental environments and planning results. (**B**) Metrics comparisons using violin plots and bar graphs. The total time is the sum of the front-end and back-end time. The total loss is the sum of the energy and time losses from the back-end optimization. The success rate is defined by whether the optimized trajectory avoids obstacles. If the trajectory avoids obstacles, the case is considered successful; otherwise, it is considered a failure.

path by the proposed model is smoother and closer to the ground truth than those generated by the other algorithms. This is because our algorithm eliminates the need for searching and directly supervises the trajectory generation process using ground truth data, thereby imitating the behavior of ground truth trajectories. Furthermore, we quantitatively analyzed the computation time, variance, and success rates of various methods. Given that the objective function of our final trajectory optimization algorithm comprised energy loss and execution time loss, the means of both losses were also measured to evaluate the quality of the final output trajectory. The front-end computation time, total planning time, overall trajectory cost, and success rate are illustrated in Fig. 2B. More detailed metrics are summarized in Table 1.

We observed that the proposed front-end algorithm maintained a high success rate and consistently generated paths with low computation time across various environments. However, other algorithms experienced an increase in computation time as the environment became more complex owing to their lack of holistic understanding and the processing of numerous irrelevant samples. Although THybrid A* uses a network to identify key areas, it fundamentally relies on discrete state space search, necessitating numerous permutations and combinations in highly constrained environments, which reduces time

Table 1. Quantitative benchmarks in various scenarios. Model time refers to the inference time of the neural network. The time ratio indicates the percentage of each method's total time relative to that of the proposed method.

	Front end					Back end				
Method	**Search time (ms)**	**Model time (ms)**	**Total time (ms)**	**Time ratio (%)**	**Variance in time (ms^2)**	**Optimization time (ms)**	**Energy cost**	**Time cost**	**Total cost**	**Success rate (%)**
Dense office										
Proposed	0†	7.193	7.193†	100.0†	1.755†	27.68†	58.32†	326.3†	384.6†	97.5†
Hybrid A*	277.1	0	277.1	3852	31232	39.92	67.64	418.3	485.9	79.6
THybrid A*	90.60	4.560†	95.16	1323	4417	35.78	62.42	391.2	453.62	93.4
Random forest										
Proposed	0†	6.896	6.896†	100.0†	1.733†	17.58†	38.18†	218.0†	256.18†	97.9
Hybrid A*	18.98	0	18.98	275.2	1105	20.58	41.10	234.0	275.1	97.7
THybrid A*	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†
Hybrid A*	156.6	0	156.6	2293	16687	32.71	63.82	364.7	428.52	95.9
THybrid A*	47.72	4.478†	52.20	764.3	1822	30.62	65.24	367.5	432.74	97.9

†Lowest value for each benchmark and scenario.

stability. Moreover, the computation time of the search algorithms varied substantially, even within a single scenario, as shown in Table 1. This instability complicated performance prediction and affected the robustness of the entire navigation system. Additionally, the experimental results demonstrated that our algorithm consistently required the least time for optimization across all scenarios and achieved the highest trajectory quality. A major reason for this is that baseline algorithms are constrained by time efficiency and memory limitations, which restrict the number of discrete samples and result in an inadequate representation of the configuration space, thus adversely affecting the solution quality.

Versatile adaptive capability

We verified the compatibility of the front end with another type of environmental description known as the terrain elevation map and extended its application to large-scale fixed-wing navigation tasks. The network took the terrain elevation map as input, leveraging topographical analysis to enhance flight safety. The simulation showed that our algorithm found a near-optimal solution in less than 10 ms, and the real-world experiment suggested that the network facilitated the safe flight of a fixed-wing aircraft.

Here, we considered a common scenario in which a fixed-wing aircraft was required to traverse mountainous and hilly terrains, ultimately reaching a target state at a given altitude, as shown in Fig. 3D. To highlight the performance of our algorithm in this problem, we compared it with the classical sampling-based algorithm RRT* (*13*), which is theoretically proven to be optimal with infinite samples. Considering the minimum turning radius constraint of the fixed-wing aircraft, we modified the connection between two states in RRT* from a straight line segment to a Dubins curve (*47*, *48*). In this case, by ensuring safety and avoiding potentially dangerous terrain, we aimed for the fixed-wing flight path not only to be smooth and short but also to select a lower altitude or flatter terrain whenever possible to achieve higher clearance from the ground. Therefore, we used two metrics to measure the path: length cost, which quantified the path length, and height cost, which measured the elevation of the ground. A smaller height cost indicates that the path traversed lower-altitude terrains, resulting in a higher relative clearance and increased safety. From the perspective of the two loss functions, the quality of the path directly generated by the proposed model, with a computation time of less than 10 ms, was comparable to that of RRT*, which required computation times that were an order of magnitude longer, depicted in Fig. 3C. Real-world experiments demonstrated that the flight path generated by the neural network served as effective waypoints to guide the collision-free traversal of fixed-wing aircraft across hills, as depicted in Fig. 3E. In particular, this approach allowed the fixed-wing platform to seamlessly cross the saddle point situated between the hill peaks, demonstrating its potential to navigate complex environments. Besides, the training environments for the real-world fixed-wing experiments were entirely synthetic. The data generation process is illustrated in the "Additional information on fixed-wing experiments" section of the Supplementary Results.

Numerically stable trajectory optimization

Although the front end rapidly generates a rough path, back-end trajectory optimization remains crucial for enhancing its quality and ensuring adherence to precise kinematic models and collision-avoidance requirements. Some existing studies (*42*) simplify the trajectory optimization problem using differential flatness, which effectively eliminates the constraints of motion equations, thereby substantially increasing computational efficiency. However, this modeling approach has inherent singularities that prevent it from guaranteeing the complete feasibility of the constraints. For example, as shown in Fig. 4B, it relied on inferring the yaw angle from the direction of the velocity, which failed when the velocity was zero, leading to numerical instability during the solution computation. In contrast, we introduced intermediate variables to smoothly remap the flat model and designed a specialized trajectory representation.

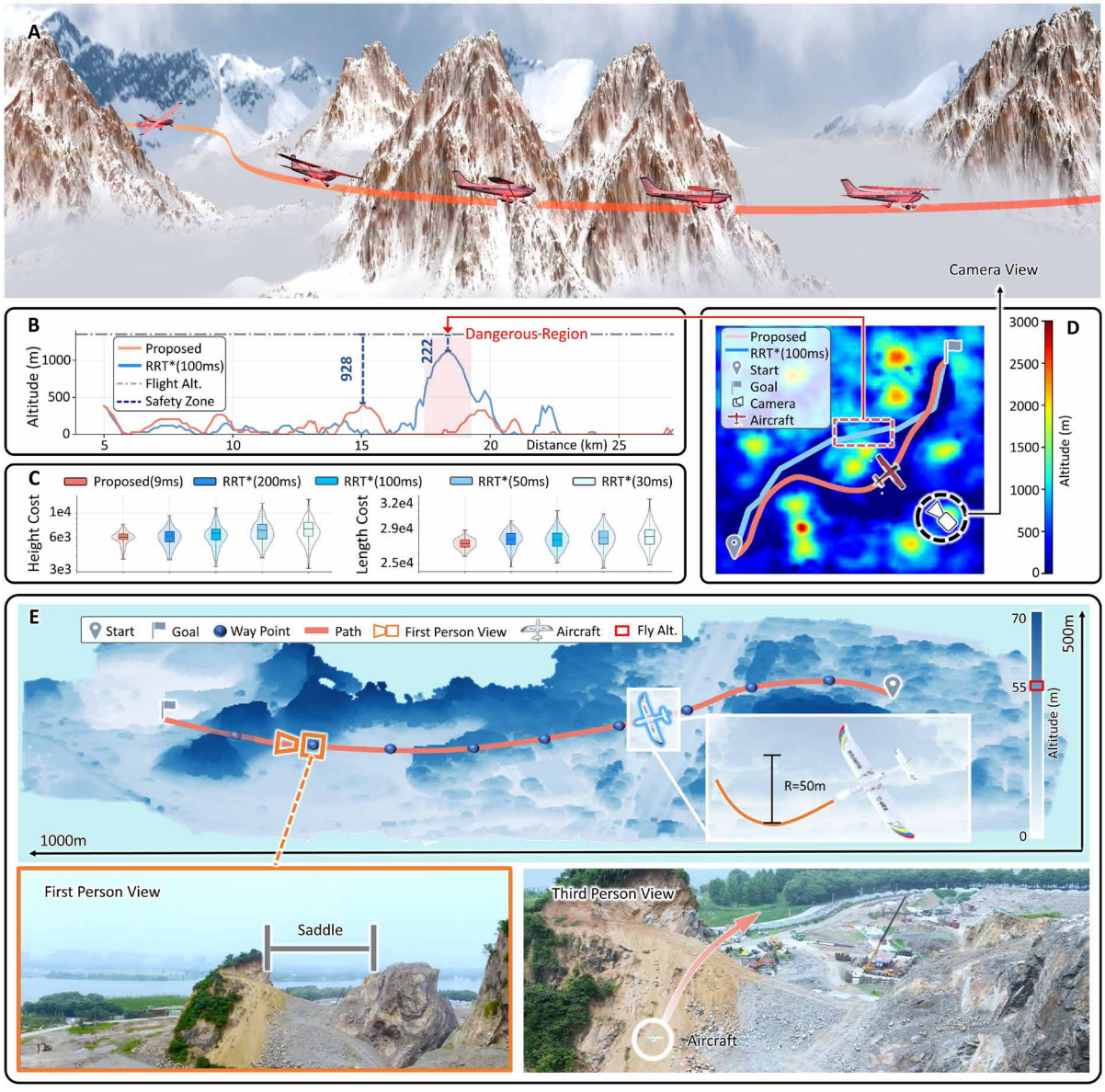

Fig. 3. Fixed-wing navigation in mountainous terrain. (**A**) Visualization of fixed-wing flight in Unity3D. (**B**) Terrain elevation at the path points. The path generated by RRT* within a limited time was suboptimal, which might cause the fixed-wing aircraft to fly too close to the terrain below, thereby making the flight unsafe. (**C**) Comparative analysis of our method and RRT* at different computation times in terms of height cost and length cost. (**D**) Terrain elevation map of the planning scene. (**E**) The 1000 m–by–50 m fixed-wing cruise experiment. The fixed-wing aircraft flew along waypoints, avoiding mountain peaks encountered during the flight. The minimum turning radius for the fixed-wing aircraft was 50 m, and the flight altitude was 55 m. A GPS was used for positioning and navigation.

This approach ensures efficient optimization and fundamentally resolves the problem, allowing stable generation of feasible solutions in highly complex environments. We rigorously validated our trajectory optimization algorithm in a challenging scenario that involved multiple forward and backward vehicle maneuvers to highlight its contributions. The results demonstrated that our algorithm achieved better numerical stability and robust generation of feasible solutions compared with a powerful baseline differential flat-based (DFB) trajectory optimization method (*42*) that has been extensively validated through experiments.

As shown in Fig. 4A, the testing environment simulated a real-life parking scenario. Here, this vehicle was instructed to depart from a designated starting point and successively approach predetermined intermediate points with prescribed yaw angles. Finally, it was required to reverse to a specified end point. From Fig. 4A, the presence of singular points led to more convoluted trajectories generated by DFB planning. However, the proposed method produced noticeably smoother trajectories. To ensure fairness, the optimization time for both methods was set to 100 ms, and the total trajectory time and the kinematic constraints were kept constant in all the methods. Given that the baseline method (*42*) faces computational singularity when the velocity is zero, we fixed it at a small value (nonsingular velocity) during forward and backward motion and analyzed the effects of this parameter. We present curves that

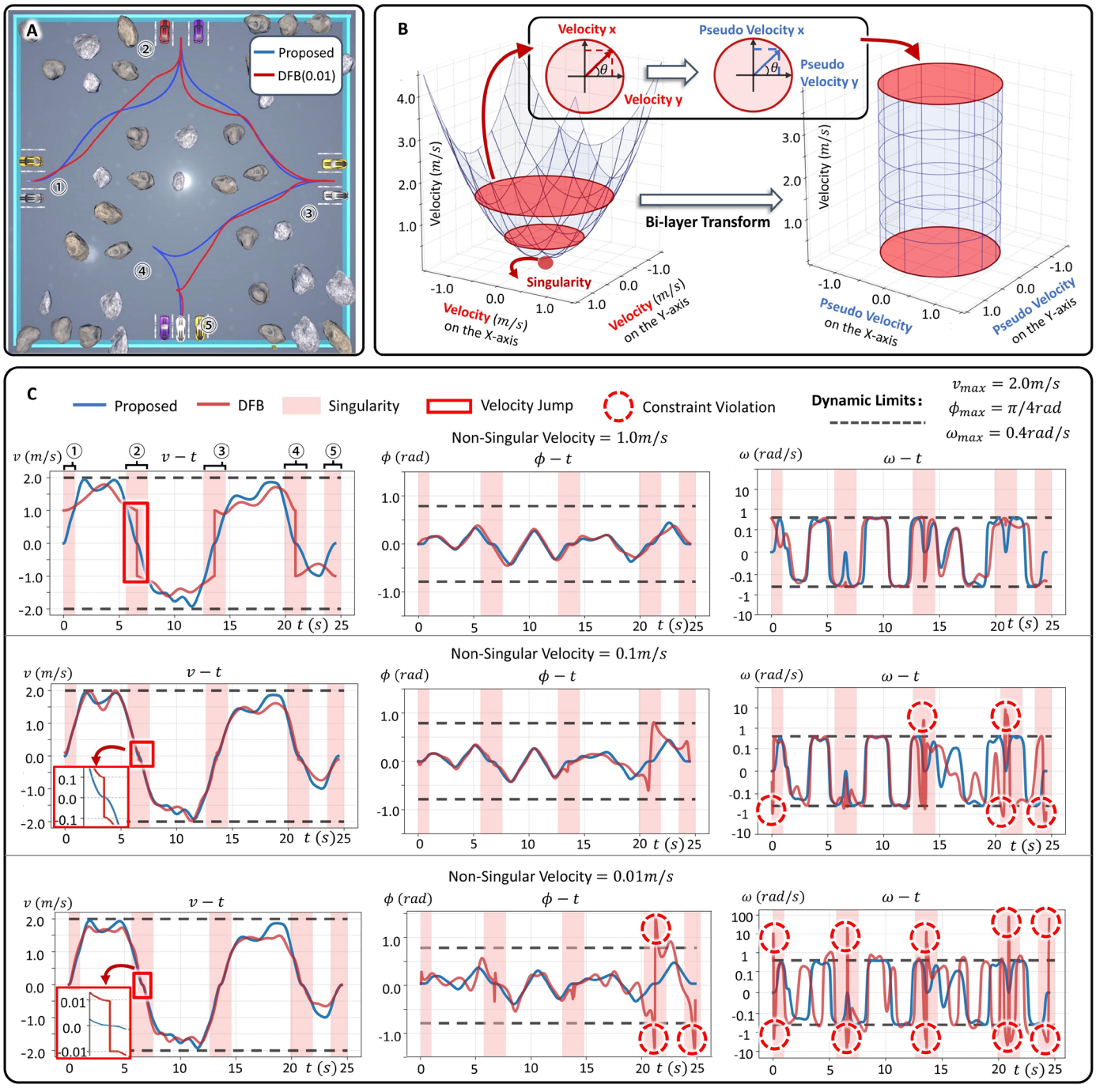

Fig. 4. Numerical stability in multiple reversing scenarios. (**A**) Visualization of planned trajectories. Here, for DFB, the switching speed during the forward and backward motions was set to 0.01 m/s to avoid computational singularities. (**B**) Visualization of singular point elimination in the flat space. Before the transformation, when the velocity of the vehicle was 0 m/s, singular points arose, preventing the calculation of the vehicle's heading angle. However, by using our bilayer transformation and introducing pseudovelocity, the magnitude of the pseudovelocity always surpassed a threshold, allowing for the recovery of the heading angle of the vehicle without encountering singularities, regardless of the actual velocity. (**C**) Comparison of velocity, steer angle, and steer angle rate curves between our trajectory and the trajectories generated by DFB under different parameters. The black dotted lines in the graphs represent the dynamic constraints.

depicted the variation of velocity v, steer angle ϕ, and steer angle rate ω of the planned trajectory over time for various cases, as shown in Fig. 4C. These curves intuitively revealed the numerical instability caused by singularities. When the nonsingular velocity was 0.1 m/s, the steer angle rate constraint of the trajectory could not be strictly satisfied. When the nonsingular velocity was 0.01 m/s, the steer angle constraint was also violated, and higher-order steer angle rates exceeded the constraints by an order of magnitude. Although a

larger nonsingular velocity set to 1.0 m/s could avoid singularity issues, it introduced discontinuity during motion, resulting in severe vibrations of mechanical components. In contrast, our method consistently exhibited good numerical stability, ensuring smooth velocity profiles and robust convergence to dynamically feasible solutions. The 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 introduced a model predictive control (MPC)–based controller to measure this error. On the basis of the statistical data provided in the "Details of the backend trajectory optimization in comparative experiments" section in Supplementary Results, the proposed method achieved a 38.2% reduction in the maximum position tracking error and a 57.7% reduction in the maximum angle tracking error compared with the baseline DFB, improving the reliability and completeness of the entire navigation system.

Large-scale outdoor maze experiment

The proposed system completed large-scale navigation tasks in a complex environment. Figure 5A depicts a vast and structurally complex maze area, approximately 50 m by 25 m in size, with multiple corners and winding passages. For the experiment, we constructed a point cloud map for the system's planning with light detection and ranging (LIDAR). Our system performed rapid and successful front-end path planning, which then served as a guide for real-time back-end optimization, calculating trajectories that considered the nonholonomic kinematics of the robot. As shown in Fig. 5B, we used Direct Drive's Diablo as the experimental platform. This platform was equipped with an NVIDIA Jetson Xavier as the onboard processor and an Ouster LIDAR with an integrated inertial measurement unit (IMU) for real-time localization (*49*).

We tested the capability of the system to rapidly plan trajectories in the complex maze. Figure 5 (C to E) illustrates the real-time position and orientation with our proposed front-end model guiding globally. Diablo quickly navigated through the lateral winding passages (15 to 40 s), straight long corridors (50 to 70 s), and vertical winding passages (90 to 120 s) of the maze, eventually reaching the destination, covering a total displacement of approximately 150 m. We designed a maze dataset with domain randomization for front-end model training (*50*, *51*), which included 60,000 maze environments, each with 50 paths. This model was deployed directly on the onboard platform, and the onboard inference time was only 25 ms. Using the network inference results as initial values, we performed real-time refinement by applying the back-end trajectory optimization method. Considering the discrepancies between the previously constructed map and the real environment, along with the potential presence of unknown obstacles, relying solely on the general guidance of the network could result in unsafe movement. As shown in Fig. 5F, the robot could swiftly avoid previously unconsidered obstacles even during high-speed motion because of our rapid back-end optimization. Figure 5G indicates that the average time for back-end replanning on the resource-constrained onboard platform is approximately 40 ms, with an average speed of 1.3 m/s for each replanned trajectory.

DISCUSSION

This study presents a hierarchical approach that integrates the spatial extraction capabilities of the neural network with the robust convergence of numerical optimization. This complementary system enables the efficient and stable generation of high-quality trajectories in diverse environments, mirroring the human intuition of initial global guidance followed by continuous adjustments.

For front-end path planning, we designed a neural network to generate paths directly from environmental data, eliminating the need for extensive sampling. The comparative experiments shown in Fig. 2 demonstrated the superior efficiency and stability of the proposed method over traditional algorithms, even in complex environments. Ablation studies showed that our image-domain network modeling reduced loss by 45% compared with the baseline, detailed in the "Ablation experiments regarding the network architecture" section of the Supplementary Results. For the back end, we designed a bilayer polynomial-based trajectory optimization to further refine the front-end output into a fully constraint-compliant and smooth trajectory. Notably, our back-end optimization algorithm combines the efficiency of flat-based methods and eliminates singularities to enhance numerical stability, ensuring reliable convergence to feasible solutions even in extreme scenarios. The experimental analysis presented in Table 1 and Fig. 4C reveals that the proposed method demonstrates stable and robust performance.

Compared with learning-based methods, traditional algorithms are resolution complete, meaning that with sufficiently high resolution, adequately considered kinematic dimensions of the robot, and an infinite number of samples, they are theoretically guaranteed to find an optimal feasible solution (*13*). However, in practical applications, the vast number of samples imposes a computational and memory burden, and the high-dimensional configuration space search or sampling leads to a combinatorial explosion. Our learning-based approach does lack theoretical proof of completeness, yet experiments showed that it performed exceptionally well in environments similar to its training set. In the "Generalization performance tests" section of the Supplementary Results, we tested the model in entirely new environments, where the success rate dropped by approximately 20%, yet it still maintained a 76.5% success rate in the worst-case scenarios. This outcome can be attributed to the network's ability to potentially extract latent features that are similar to those in the training set, offering the possibility of accurately interpreting the environment even in unfamiliar settings.

The primary challenge in real-world deployment arises from the domain shift between simulation and reality. First, unlike the clean and deterministic conditions of simulation, real-world environments are inherently more complex and diverse, often exhibiting various uncertainties such as sensor noise and map occlusions. These factors can obscure or distort the actual geometry of the environment, making it much more difficult for the planner to identify feasible paths. Besides, such uncertainties may even arise in unforeseen ways, causing the robot's current observations to fall outside the training distribution and ultimately degrading model performance. Second, unexpected corner cases may occur in real-world scenarios where the distribution of obstacle shapes and structures differs substantially from that in simulation, potentially exposing the model to map configurations it has not encountered during training and further degrading its performance. To narrow the sim-to-real gap, we used domain randomization (*51*–*53*) to augment and diversify the training data, thereby enhancing the robustness and generalization capability of the network. Furthermore, future work will focus on developing higher-fidelity scene simulators (*54*) and incorporating multimodal semantic

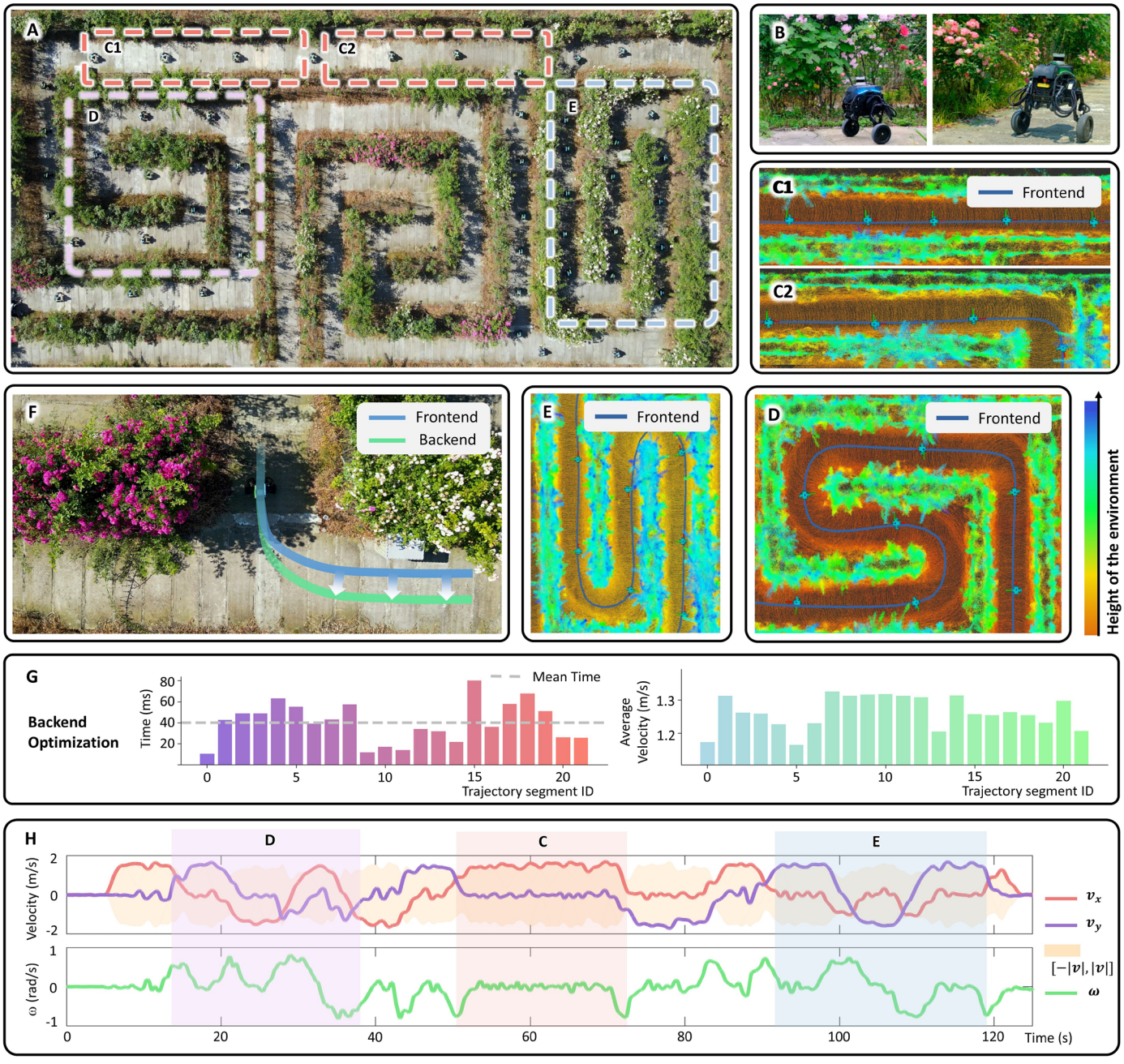

Fig. 5. Large-scale outdoor maze experiment. (**A**) Sequential snapshots showing Diablo navigating through a complex maze. (**B**) Snapshots of Diablo. (**C**) A long straight corridor. (**D**) A lateral winding maze. (**E**) A vertical winding maze. (**F**) A schematic diagram of back-end optimization. (**G**) Bar chart showing the results of trajectory optimization. (**H**) States of the robot during maze navigation.

information into the planner to enhance environmental understanding and robustness to inaccuracies.

In application, our planner is goal-directed, making it inherently suited to serve as a foundational module for task-oriented systems. This characteristic allows it to be easily integrated into various higher-level applications, such as autonomous exploration, precise tracking, and complex search and rescue missions, in which the target points are provided by a decision-making module, thereby enabling seamless coordination and execution of the tasks.

The proposed approach does have some limitations. Careful adjustment of the weights assigned to multiple losses is required, which can be a time-consuming process. Moreover, our model currently lacks consideration for multimodal characteristics, and to mitigate the risk of falling into infeasible local optima that this issue may cause, we devised the following strategies. First, we improved the optimality of the ground truth data used for supervision through high-resolution search or sampling and further back-end optimization. Second, during training, we introduced task-specific unsupervised

losses, such as nonholonomic kinematics loss and obstacle avoidance loss, to guide the paths away from local minima and further distance them from obstacles. Third, we structured our method as a hierarchical framework. The proposed efficient and robust back-end optimization refines paths into high-quality feasible trajectories in most cases. Enhancing multimodality is beneficial for exploring the solution space more thoroughly, thereby improving global optimality. A promising improvement would be to incorporate advanced generative models (*26*, *27*, *55*). However, these methods may face a trade-off between generation quality and computational efficiency and often rely on sufficient training samples because of the need for a detailed understanding and modeling of data distributions. Future advancements from the artificial intelligence (AI) community and hardware (*56*) hold great potential to address these issues, which is also one of our research goals moving forward.

MATERIALS AND METHODS

Neural path planning

Problem encoding

Our learning-based path planner inputs the navigation start and goal state $(\mathbf{x}_1, \mathbf{x}_N)$, as well as the grid-based environment ε of size $H \times W$, and directly outputs a path $p = \{\mathbf{x}_1^p, \ldots, \mathbf{x}_i^p, \ldots, \mathbf{x}_N^p\}$ 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 in two-dimensional (2D) space (*57*). Furthermore, ε is typically represented by a Euclidean signed distance field (ESDF) (*58*) where each element represents the signed distance from obstacles at that location. Inspired by the work (*19*), we adopted a start-goal encoding strategy by highlighting patches on a tensor of size $H \times W$. Specifically, we assigned 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 were set as 0. To fully represent the $\mathbb{SE}(2)$ space, we introduced two additional $H \times W$ 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, whereas the other tensor represents the sine values. In particular, the aforementioned representations of the environment and the start and goal $\mathbb{SE}(2)$ states are concatenated to form a $4 \times H \times W$ tensor shown in Fig. 6A, which serves as the input to our network.

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), as illustrated in Fig. 6. In practice, directly localizing the position of a specific state point within the environment is challenging and labor intensive (*59*). Therefore, drawing inspiration from the region-based convolutional neural network (R-CNN) (*60*), our network follows a two-stage inference architecture. Initially, ε is uniformly partitioned into $H_l \times W_l$ region proposals, where each region proposal corresponds to a block of size $\frac{H}{H_l} \times \frac{W}{W_l}$ in the environment. The center point of each region proposal is designated as an anchor point. Subsequently, we used GDL 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, by leveraging the LCL, we further obtained the accurate position of the state point on the basis of the environmental features and the probability distribution obtained earlier. Intuitively, we aimed 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.

Computation stream

First, Fig. 6B illustrates the use of the FEL to encode the path plan problem into a high-dimensional latent space, and the size of this latent feature $\mathcal{M}$ is downsampled to $d \times H_l \times W_l$ to match the shape of the probability distribution of the region proposals. Here, d represents the user-defined dimension of the latent features. Figure 6C demonstrates the subsequent input of $\mathcal{M}$ into the GDL, resulting in the probability distribution map $\mathcal{P}$ over the region proposals for N points along the path, with a size of $N \times H_l \times W_l$. Here, we denote the probability of the ith point belonging to the (j, k)th region as $\varrho_{i,j,k}$. Moreover, it is evident that these probabilities satisfy the principle of probability normalization

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

The probability distribution $\mathcal{P}$ and latent features $\mathcal{M}$ are concatenated and fed into the LCL, which produces a tensor $\mathcal{O}$ of size $3N \times H_l \times W_l$, as shown in Fig. 6D. The physical meaning of this tensor $\mathcal{O}$ is to assign an orientation angle $b_{i,j,k}^{\theta}$ and positional offsets $\left(b_{i,j,k}^{x}, b_{i,j,k}^{y}\right)$ relative to the anchor for each region. Instinctively, we selected the region with the highest probability from $\mathcal{P}$ and obtained 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 nondifferentiable, posing challenges for training. Consequently, we used a weighted summation based on $\mathcal{P}$ to compute the state of any point, thus enabling differentiability and facilitating effective training

$$r_{x,i}^p = \hat{r}_{x,i}^p + \tilde{r}_{x,i}^p \tag{2}$$

$$\hat{r}_{x,i}^p = \sum_{j=0}^{H_l-1} \sum_{k=0}^{W_l-1} \varrho_{i,j,k}\left(j+0.5\right) r_s, \tilde{r}_{x,i}^p = \sum_{j=0}^{H_l-1} \sum_{k=0}^{W_l-1} \varrho_{i,j,k} b_{i,j,k}^x$$

$$r_{y,i}^p = \hat{r}_{y,i}^p + \tilde{r}_{y,i}^p \tag{3}$$

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

$$\theta_i^p = \sum_{j=0}^{H_l-1} \sum_{k=0}^{W_l-1} \varrho_{i,j,k} b_{i,j,k}^{\theta}, \forall i \in \{1, \ldots, N\} \tag{4}$$

where $(r_{x,i}^p, r_{y,i}^p)$ and θ_i^p denote the position and orientation angle of the ith state point, respectively. $\hat{r}_{x,i}^p$ and $\hat{r}_{y,i}^p$ denote the weighted sum

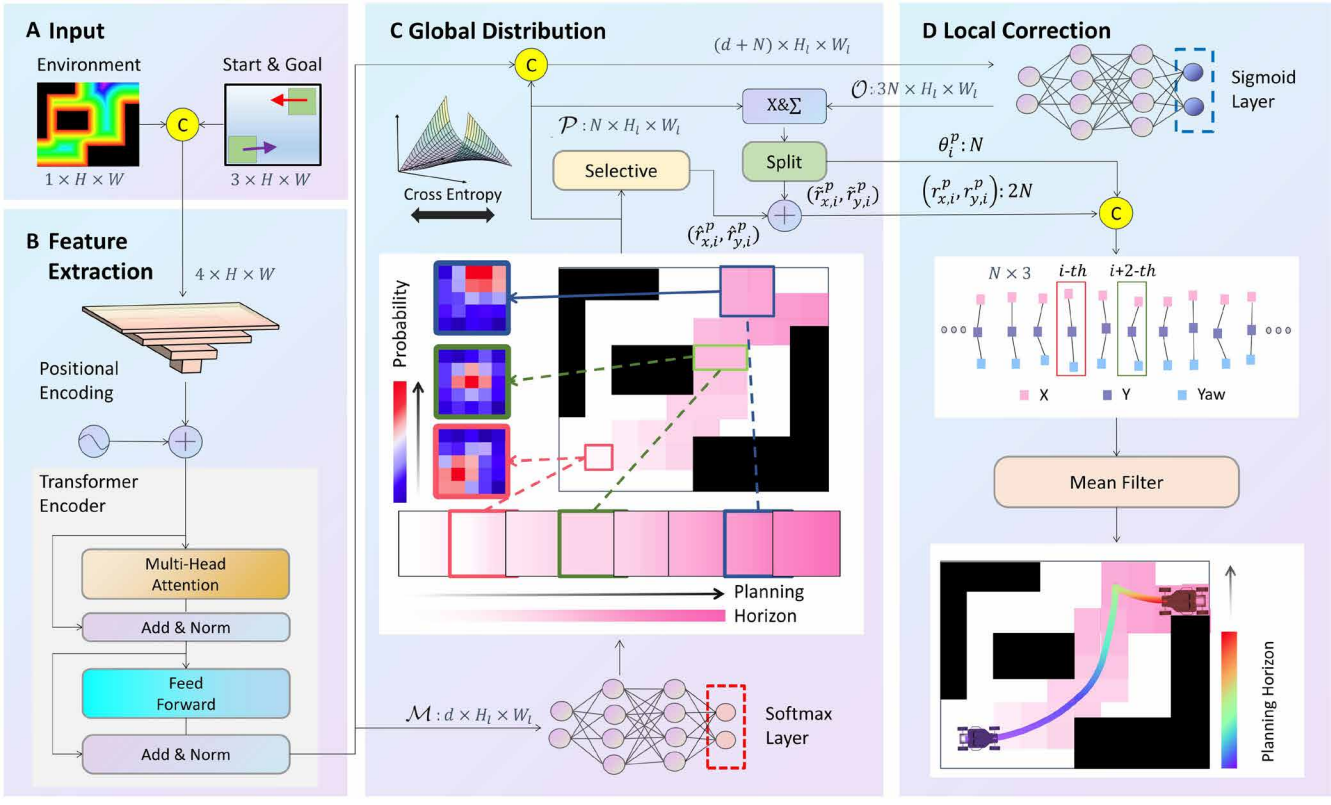

Fig. 6. Front-end architecture. The yellow circles containing "C" represent concatenation operators. (**A**) Input of the network. (**B**) FEL, where a transformer encoder extracts high-level features. (**C**) GDL. This layer infers a probability distribution over the path based on the geometric structure of the environment. The selection module implies the weighted computation of the anchor positions based on the probability distribution $\mathcal{P}$. (**D**) LCL. This layer generates the position and yaw of the path from the predicted distribution.

of anchor positions based on the probability distribution $\mathcal{P}$, which can also be regarded as the output of the GDL. Here, we define the rough path comprising $\left(\hat{r}_{x,i}^{p}, \hat{r}_{y,i}^{p}\right)$ as $\hat{p}$ for simplicity. On the other hand, $\tilde{r}_{x,i}^{p}$ and $\tilde{r}_{y,i}^{p}$ refer to the weighted sum of positional offsets based on $\mathcal{P}$, which aim to correct $\hat{p}$ and improve its overall quality to get p. Moreover, to mitigate outliers and increase smoothness, we used a sliding window to filter the mean along the aforementioned path p.

As illustrated in Fig. 1, we desired to plan a path that can learn the behavior policy of the ground truth and satisfy the fundamental constraints of nonholonomic motion and obstacle avoidance. Therefore, our loss function $\mathcal{L}$ comprises supervised terms that mimic the ground truth and unsupervised terms that purely measure the validity of the path

$$\begin{aligned}\mathcal{L} = {} & w_{\text{ce}}\mathcal{L}_{\text{ce}} + w_{\text{mse}}\mathcal{L}_{\text{mse}} + w_{\text{smo}}\mathcal{L}_{\text{smo}} + \\ & w_{\text{hol}}\mathcal{L}_{\text{hol}} + w_{\text{cur}}\mathcal{L}_{\text{cur}} + w_{\text{uni}}\mathcal{L}_{\text{uni}} + w_{\text{obs}}\mathcal{L}_{\text{obs}}\end{aligned} \tag{5}$$

where w_* denotes the weight corresponding to each loss. These customized unsupervised loss terms enable the network to better uncover latent causal relationships in the training data (*61*). Here, $\mathcal{L}_{\text{ce}}$ denotes the loss in anchor point classification. Because we modeled the role of the GDL as a multiclassification problem, $\mathcal{L}_{\text{ce}}$ is used to maximize the probability of the region containing the ground truth point. $\mathcal{L}_{\text{mse}}$ is the path supervision loss, which is used to supervise the mean square error between the $\mathbb{SE}(2)$ states of the points on the output path and their corresponding ground truth values. $\mathcal{L}_{\text{smo}}$ denotes the smoothness loss used to enhance the smoothness of the planned path. $\mathcal{L}_{\text{hol}}$ denotes the nonholonomic dynamic loss that emphasizes the alignment between the direction of the line connecting adjacent points and the direction of the orientation angle. $\mathcal{L}_{\text{cur}}$ denotes the loss of curvature constraint that penalizes exceeding the specified curvature threshold. $\mathcal{L}_{\text{uni}}$ denotes the uniform loss, which is applied to encourage a more uniform spatial distribution of points along the path. This is achieved by penalizing the variance of the positions of the points in the space. $\mathcal{L}_{\text{obs}}$ denotes the obstacle avoidance loss, which is used to penalize potential collisions with obstacles. The specific rigorous mathematical expressions and corresponding weights for each loss can be found in the "Parameters and experimental details for benchmarks in front end path planning" section in the Supplementary Results.

Bilayer trajectory optimization

Remapping of differential flat models

The characteristic of differentially flat systems is their ability to analytically represent the state of a system using a combination of flat outputs and their finite-dimensional derivatives. Here, we considered the example of Ackermann kinematics (*62*), as shown in Fig. 7A. By selecting the flat outputs $\sigma := \left(p_x, p_y\right)^{\mathrm{T}}$ as the position at the center

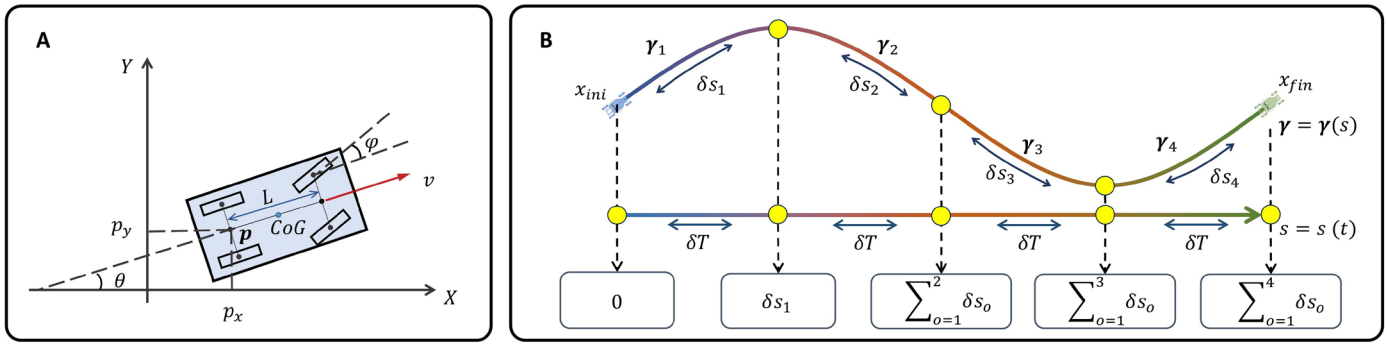

Fig. 7. Back-end mechanism. (**A**) Ackermann vehicle model. (**B**) Visualization of bilayer trajectory representation. Flat output $\boldsymbol{\gamma}$ is represented by a piecewise polynomial parameterized by the pseudo arc s. Similarly, pseudo arc s is also represented by a piecewise polynomial parameterized by time t.

of the rear wheels, the other high-order states of the robot during forward motion can be expressed as follows

$$v = \|\dot{\sigma}_{|t}\|_2, \theta = \arctan2\left(\dot{\sigma}_{y|t}, \dot{\sigma}_{x|t}\right) \tag{6}$$

$$a = \frac{\ddot{\sigma}_{|t}^{\mathrm{T}}\dot{\sigma}_{|t}}{\|\dot{\sigma}_{|t}\|_2}, \phi = \arctan\left(\frac{\ddot{\sigma}_{|t}^{\mathrm{T}}\mathrm{B}\dot{\sigma}_{|t}L}{\|\dot{\sigma}_{|t}\|_2^3}\right) \tag{7}$$

$$\omega = L\frac{\dddot{\sigma}_{|t}^{\mathrm{T}}\mathrm{B}\dot{\sigma}_{|t}\|\dot{\sigma}_{|t}\|_2^3 - 3\ddot{\sigma}_{|t}^{\mathrm{T}}\mathrm{B}\dot{\sigma}_{|t}\ddot{\sigma}_{|t}^{\mathrm{T}}\dot{\sigma}_{|t}\|\dot{\sigma}_{|t}\|_2}{\|\dot{\sigma}_{|t}\|_2^6 + \left(\ddot{\sigma}_{|t}^{\mathrm{T}}\mathrm{B}\dot{\sigma}_{|t}L\right)^2} \tag{8}$$

where $\dot{\sigma}_{|t}$, $\ddot{\sigma}_{|t}$, and $\dddot{\sigma}_{|t}$ denote the first, second, and third derivatives of the flat output with respect to time, respectively. $B \coloneqq \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ denotes an auxiliary antisymmetric matrix for computational convenience. Besides, v is the longitudinal velocity with respect to the body frame of the vehicle, a denotes the longitudinal acceleration, ϕ denotes the steering angle of the front wheels, ω denotes the steer angular velocity, and L denotes the wheelbase.

Although leveraging differential flatness can accelerate the optimization process, it is noteworthy that when the velocity approaches zero, certain states such as Eqs. 7 and 8 can encounter singularities. However, we introduced a pseudo arc to indirectly derive the flat model that fundamentally eradicates the singularity issues

$$\sigma = \gamma(s), s = s(t) \tag{9}$$

where $s \in \mathbb{R}^+$ is the pseudo arc. Then, based on the scale-time mapping, the finite-dimensional derivative of the flat output can also be derived as follows

$$\dot{\sigma}_{|t} = \dot{\gamma}_{|s}\dot{s}_{|t} \tag{10}$$

$$\ddot{\sigma}_{|t} = \dot{\gamma}_{|s}\ddot{s}_{|t} + \dot{s}_{|t}^2\ddot{\gamma}_{|s} \tag{11}$$

$$\dddot{\sigma}_{|t} = \dddot{s}_{|t}\dot{\gamma}_{|s} + 3\dot{s}_{|t}\ddot{\gamma}_{|s}\ddot{s}_{|t} + \dot{s}_{|t}^3\dddot{\gamma}_{|s} \tag{12}$$

where $\dot{\gamma}_{|s}$, $\ddot{\gamma}_{|s}$, and $\dddot{\gamma}_{|s}$ denote the first, second, and third derivatives of the flat output with respect to pseudo arc, respectively. Notably, $\dot{\sigma}_{|t}$ denotes the actual motion velocity of the robot, and $\dot{\gamma}_{|s}$ is defined as the pseudovelocity. Moreover, the physical meaning of $\dot{s}_{|t} \geq 0$ is the magnitude of velocity along the pseudo arc.

We substituted Eqs. 10 to 12 in Eqs. 6 to 8, thus modifying the differential flatness model. Taking ϕ and ω as examples, they are redefined as follows

$$\phi = \arctan\left(\frac{\ddot{\gamma}_{|s}^{\mathrm{T}}\mathrm{B}\dot{\gamma}_{|s}L}{\|\dot{\gamma}_{|s}\|_2^3}\right) \tag{13}$$

$$\omega = L\frac{\dddot{\gamma}_{|s}^{\mathrm{T}}\mathrm{B}\dot{\gamma}_{|s}\|\dot{\gamma}_{|s}\|_2^3 - 3\ddot{\gamma}_{|s}^{\mathrm{T}}\mathrm{B}\dot{\gamma}_{|s}\ddot{\gamma}_{|s}^{\mathrm{T}}\dot{\gamma}_{|s}\|\dot{\gamma}_{|s}\|_2}{\|\dot{\gamma}_{|s}\|_2^6 + \left(\ddot{\gamma}_{|s}^{\mathrm{T}}\mathrm{B}\dot{\gamma}_{|s}L\right)^2}\dot{s}_{|t} \tag{14}$$

When the robot velocity is zero, we can set $\dot{s}_{|t}$ to zero and keep pseudovelocity $\dot{\gamma}_{|s}$ as nonzero. Consequently, the denominators in Eqs. 13 and 14 are strictly positive, thereby eliminating the original singularity point. Next, we provided a detailed description of the parametric form of $\sigma(t) = \gamma(s(t))$ based on bilayer piecewise polynomials.

Bilayer polynomial-based trajectory representation

To ensure sufficient degrees of freedom and smoothness, we parameterized $\boldsymbol{\gamma}(s)$ and $s(t)$ using piecewise polynomials, thereby establishing a compact nonlinear optimization problem. We formulated the trajectory $\boldsymbol{\gamma}(s)$ as an M-piece polynomial with degree $D = 2u - 1$, which is parameterized by pseudo arc $\delta\mathbf{s} = \left(\delta s_1, \ldots, \delta s_M\right)^{\mathrm{T}} \in \mathbb{R}^M$ corresponding to each piece and coefficient matrix $\mathbf{c}^p = \left(\left(\mathbf{c}_1^p\right)^{\mathrm{T}}, \ldots, \left(\mathbf{c}_M^p\right)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2Mu\times 2}$. Besides, u denotes the control dimension. Similarly, pseudo arc $s(t)$ is represented as a 1D and time-uniform M-piece polynomial, parameterized by the time interval for each piece δT and coefficient matrix $\mathbf{c}^s = \left(\left(\mathbf{c}_1^s\right)^{\mathrm{T}}, \ldots, \left(\mathbf{c}_M^s\right)^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2Mu}$. Moreover, we considered a strict correspondence between each piece of the bilayer piecewise polynomials such that for every time interval δT, the robot should traverse the corresponding pseudo arc

$$s(i^*\delta T) = \sum_{j=1}^{i}\delta s_j, \forall i \in \{1, 2, 3, \ldots, M\} \tag{15}$$

On the basis of the above modeling, the ith piece of $\boldsymbol{\gamma}_i$ is represented as follows

$$\gamma_i(s_i) := \left(\mathbf{c}_i^p\right)^{\mathrm{T}} \beta\left(s_i - \sum_{j=1}^{i-1} \delta s_j\right)$$
$$s_i(t) := \left(\mathbf{c}_i^s\right)^{\mathrm{T}} \beta(t), \forall t \in [0, \delta T] \tag{16}$$

where $\beta(x) := \left(1, x, x^2, \ldots, x^N\right)^{\mathrm{T}}$ denotes a natural basis function. Moreover, due to the strict correspondence between each piece of the bilayer polynomials, we could further derive the following equation from Eq. 15

$$s_i(0) := \sum_{j=1}^{i-1} \delta s_j, s_i(\delta T) := s_i(0) + \delta s_i \tag{17}$$

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

$$\sigma(t) = \gamma(s(t)) = \gamma_i(s(t)), s(t) = s_i(t - (i-1) * \delta T),$$
$$t \in \left[(i-1) * \delta T, i^* \delta T\right] \tag{18}$$

To facilitate an intuitive understanding, we visualized the trajectory representation of the bilayer polynomial, as illustrated in Fig. 7B. With motion feasibility constraints, the minimum control effort problem based on the modified flatness model in Eqs. 13 and 14, incorporating first-order temporal regularization, is formulated as a nonlinear constrained optimization

$$\min_{c^p, c^s, \delta s, \delta T \in \mathbb{R}^+} J = \int_0^{T_s} \sigma_{|t}^{(u)}(t)^{\mathrm{T}} \sigma_{|t}^{(u)}(t) dt + \rho T_s \tag{19}$$

subject to

$$\mathcal{B}\left(\sigma_{|t}(0) \ldots \sigma_{|t}^{(u-1)}(0), \sigma_{|t}\left(T_s\right) \ldots \sigma_{|t}^{(u-1)}\left(T_s\right)\right) = 0 \tag{20}$$

$$\mathcal{T}\left(\gamma_1 \ldots \gamma_M, s_1 \ldots, s_M, \delta s, \delta T\right) = 0 \tag{21}$$

$$\left\|\dot{\gamma}_s(s(t))\right\|_2 > \alpha \tag{22}$$

$$\dot{s}_{|t}(t) \geq 0 \tag{23}$$

$$\mathcal{G}\left(\gamma(s(t)), \ldots, \gamma_{|s}^{(u)}(s(t)), s(t), \ldots, s_{|t}^{(u)}(t)\right) \preceq 0 \tag{24}$$

where $\rho \in \mathbb{R}^+$ denotes a user-defined weight for the time regularization term to restrict the total duration T_s. Equation 20 is the boundary condition representing the initial and final state constraints of the trajectory. Equation 21 is the continuity constraint at the junctions of the piecewise polynomials. Equation 22 is a minimum pseudovelocity constraint introduced to avoid singularities, where α is a threshold value. Equation 23 is the positive-definite constraint on pseudovelocity. $\mathcal{G}$ encompasses the common inequality constraints considered in trajectory planning problems, including dynamic feasibility and obstacle avoidance constraints. To solve the constrained nonlinear optimization problem, we applied the minimum energy condition of the trajectory (*43*) to reformulate the original problem as Eqs. 19 to 24 and eliminated its equality constraints in Eqs. 20 and 21 without sacrificing optimality. Subsequently, we used the augmented Lagrange multiplier method (*63*) to relax the inequality constraints in Eqs. 22 to 24. This method iteratively updates the dual variables while solving the approximate unconstrained problem using the highly effective limited-memory Broyden-Fletcher-Goldfarb-Shanno algorithm (*64*). Furthermore, to enhance both the efficiency and accuracy of the solution, we analytically derived the gradients of the objective function and constraints with respect to the optimization variables. Detailed solution procedures are presented in the "Gradient backpropagation chain trajectory optimization" section of the Supplementary Methods.

Statistical analysis

We applied several statistical visualization methods to assess and compared experimental results: box and violin plots (Figs. 2B and 3C), line charts (Figs. 3B and 4C), and bar graphs (Figs. 2B and 5G). The violin and box plots were overlaid to jointly convey both the detailed distribution and summary statistics of each group. The violin plots represent the probability density of the data via kernel density estimation, with the width indicating the concentration of values at different levels. Line charts were used to illustrate the temporal evolution of key variables. Bar charts display the mean performance across conditions. Vertical error bars in Fig. 2B represent the 95% confidence interval (CI) of the mean. To compute the CI, we first calculated the SEM and then applied the standard normal approximation to estimate the 95% CI as

$$P\left(\bar{x} - z_{1-\alpha/2} \cdot \frac{s}{\sqrt{n}} \leq \mu \leq \bar{x} + z_{1-\alpha/2} \cdot \frac{s}{\sqrt{n}}\right) = 1 - \alpha \tag{25}$$

with $\alpha = 0.05$ and $z_{1-\alpha/2} \approx 1.96$. Here, $\bar{x}$ is the sample mean, s is the sample SD, and $n = 1000$ is the sample size.

Supplementary Materials

The PDF file includes:

Methods
Results
Tables S1 to S6
Figs. S1 to S9
Discussion
Legends for movies S1 to S5
References (*65–70*)

Other Supplementary Material for this manuscript includes the following:

Movies S1 to S5

REFERENCES AND NOTES

1. E. A. Capaldi, G. E. Robinson, S. E. Fahrbach, Neuroethology of spatial learning: The birds and the bees. *Annu. Rev. Psychol.* **50**, 651–682 (1999).
2. K. M. I. Chu, S. H. Seto, I. N. Beloozerova, V. Marlinski, Strategies for obstacle avoidance during walking in the cat. *J. Neurophysiol.* **118**, 817–831 (2017).
3. 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 navigation: A synthesis" in *Animal Thinking: Contemporary Issues in Comparative Cognition* (MIT Press, 2011), pp. 51–76.
4. R. Alterovitz, S. Koenig, M. Likhachev, Robot planning in the real world: Research challenges and opportunities. *AI Mag.* **37**, 76–84 (2016).

5. 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).
6. M. I. Sereno, R.-S. Huang, A human parietal face area contains aligned head-centered visual and tactile maps. *Nat. Neurosci.* **9**, 1337–1343 (2006).
7. B. A. Wandell, S. O. Dumoulin, A. A. Brewer, Visual field maps in human cortex. *Neuron* **56**, 366–383 (2007).
8. A. D. Ekstrom, S. Y. Bookheimer, Spatial and temporal episodic memory retrieval recruit dissociable functional networks in the human brain. *Learn. Mem.* **14**, 645–654 (2007).
9. S. K. Debnath, R. Omar, N. B. A. Latip, S. Shely, E. Nadira, C. K. N. C. K. Melor, T. K. Chakraborty, E. Natarajan, A review on graph search algorithms for optimal energy efficient path planning for an unmanned air vehicle. *Indones. J. Electr. Eng. Comput. Sci.* **15**, 743–749 (2019).
10. A.Bry, N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proceedings of the IEEE International Conference on Robotics and Automation* (IEEE, 2011), pp. 723–730.
11. D. Dolgov, S. Thrun, M. Montemerlo, J. Diebel, Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Robot. Res.* **29**, 485–501 (2010).
12. 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," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2014), pp. 2997–3004.
13. S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **30**, 846–894 (2011).
14. R. Pepy, A. Lambert, H. Mounier, "Path planning using a dynamic vehicle model," in *2006 2nd International Conference on Information & Communication Technologies* (IEEE, 2006), vol. 1, pp. 781–786.
15. M. Phillips, M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2011), pp. 5628–5635.
16. Z. Ren, S. Rathinam, M. Likhachev, H. Choset, Multi-objective safe-interval path planning with dynamic obstacles. *IEEE Robot. Autom. Lett.* **7**, 8154–8161 (2022).
17. D. J. Webb, J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation* (IEEE, 2013), pp. 5054–5061.
18. J. Huh, D. D. Lee, 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, 2011), pp. 5772–5779.
19. 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:2106.02791 [cs.RO] (2021).
20. J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, 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.
21. D. Kim, K. Huh, Neural motion planning for autonomous parking. *Int. J. Control Autom. Syst.* **21**, 1309–1318 (2023).
22. 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 Robot. Autom. Lett.* **6**, 4496–4503 (2021).
23. A. H. Qureshi, A. Simeonov, M. J. Bency, M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA) (IEEE, 2019)*, pp. 2118–2124.
24. J. Wang, X. Jia, T. Zhang, N. Ma, M. Q.-H. Meng, Deep neural network enhanced sampling-based path planning in 3D space. *IEEE Trans Autom Sci Eng* **19**, 3434–3443 (2022).
25. R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, A. Kanezaki, "Path planning using Neural A* search," in *Proceedings of the 38th International Conference on Machine Learning* (MLResearchPress, 2021), pp. 12029–12039.
26. J. Ho, A. Jain, P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin, Eds. (Curran Associates, 2020), pp. 6840–6851.
27. C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, S. Song, Diffusion policy: Visuomotor policy learning via action diffusion. *Intl. J. Robot. Res.*, 10.1177/02783649241273668 (2024).
28. S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, S.-C. Zhu, "Diffusion-based generation, optimization, and planning in 3D scenes," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2023), pp. 16750–16761.
29. J. Carvalho, A. T. Le, M. Baierl, D. Koert, J. Peters, "Motion planning diffusion: Learning and planning of robot motions with diffusion models," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2023), pp. 1916–1923.
30. J. Liu, M. Stamatopoulou, D. Kanoulas, "Dipper: Diffusion-based 2D path planner applied on legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2024), pp. 9264–9270.
31. G. Zhao, T. Wu, Y. Chen, F. Gao, Learning speed adaptation for flight in clutter. *IEEE Robot. Autom. Lett.* **9**, 7222–7229 (2024).
32. X. Zhou, Z. Wang, H. Ye, C. Xu, F. Gao, EGO-planner: An ESDF-free gradient-based local planner for quadrotors. *IEEE Robot. Autom. Lett.* **6**, 478–485 (2021).
33. A. D. Ekstrom, P. F. Hill, Spatial navigation and memory: A review of the similarities and differences relevant to brain models and age. *Neuron* **111**, 1037–1049 (2023).
34. A. D. Ekstrom, E. A. Isham, Human spatial navigation: Representations across dimensions and scales. *Curr. Opin. Behav. Sci.* **17**, 84–89 (2017).
35. W. Ding, L. Zhang, J. Chen, S. Shen, Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor. *IEEE Robot. Autom. Lett.* **4**, 2997–3004 (2019).
36. W. Lim, S. Lee, M. Sunwoo, K. Jo, Hybrid trajectory planning for autonomous driving in on-road dynamic scenarios. *IEEE Trans. Intell. Transp. Syst.* **22**, 341–355 (2019).
37. K. Bergman, 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.
38. B. Li, Z. Shao, A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowl. Based Syst.* **86**, 11–20 (2015).
39. S. Shi, Y. Xiong, J. Chen, C. Xiong, A bilevel optimal motion planning (BOMP) model with application to autonomous parking. *Int. J. Intell. Robot. Appl.* **3**, 370–382 (2019).
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 Trans. Intell. Transp. Syst.* **23**, 3724–3736 (2021).
41. X. Zhang, A. Liniger, A. Sakai, F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)* (IEEE, 2018), pp. 4327–4332.
42. 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 Trans. Intell. Transp. Syst.* **25**, 1797–1814 (2024).
43. Z. Wang, X. Zhou, C. Xu, F. Gao, Geometrically constrained trajectory optimization for multicopters. *IEEE Trans. Robot.* **38**, 3259–3278 (2022).
44. B. J. Cohen, S. Chitta, M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE International Conference on Robotics and Automation* (IEEE, 2010), pp. 2902–2908.
45. M. Pivtoraiko, A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IEEE, 2011), pp. 2172–2179.
46. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, Eds. (Curran Associates, 2017).
47. I. Lugo-Cárdenas, G. Flores, S. Salazar, R. Lozano, "Dubins path generation for a fixed wing UAV," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)* (IEEE, 2014), pp. 339–346.
48. J. Reeds, L. Shepp, Optimal paths for a car that goes both forwards and backwards. *Pac. J. Math.* **145**, 367–393 (1990).
49. W. Xu, Y. Cai, D. He, J. Lin, F. Zhang, FAST-LIO2: Fast direct LiDAR-inertial odometry. *IEEE Trans. Robot.* **38**, 2053–2073 (2022).
50. J. B. Kruskal Jr., On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. Am. Math. Soc.* **7**, 48–50 (1956).
51. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2017), pp. 23–30.
52. T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.* **7**, eabk2822 (2022).
53. Y. Song, A. Romero, M. Müller, V. Koltun, D. Scaramuzza, Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. *Sci. Robot.* **8**, eadg1462 (2023).
54. M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, D. Batra, "Habitat: A platform for embodied AI research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (IEEE, 2019).
55. J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models. arXiv:2010.02502 [cs.LG] (2020).
56. G. Moore, Cramming more components onto integrated circuits (1965).
57. R. M. Murray, Z. Li, S. Shankar Sastry, *A Mathematical Introduction to Robotic Manipulation* (CRC Press, 2017).
58. L. Han, F. Gao, B. Zhou, S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 4423–4430.

59. K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.
60. R. Girshick, J. Donahue, T. Darrell, J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2014), pp. 580–587.
61. F. Yang, C. Wang, C. Cadena, M. Hutter, iPlanner: Imperative path planning. arXiv:2302.11434 [cs.RO] (2023).
62. M. Tanelli, M. Corno, S. Saveresi, *Modelling, Simulation and Control of Two-Wheeled Vehicles* (John Wiley & Sons, 2014).
63. R. Tyrrell Rockafellar, Augmented lagrange multiplier functions and duality in nonconvex programming. *SIAM J. Control* **12**, 268–285 (1974).
64. D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**, 503–528 (1989).
65. P. Gabrovšek, Analysis of maze generating algorithms. *IPSI Trans. Internet Res.* **15**, 23–30 (2019).
66. P. H. Kim, "Intelligent maze generation," thesis, Ohio State University, Columbus, OH (2019).
67. A. Kozlova, J. A. Brown, E. Reading, "Examination of representational expression in maze generation algorithms," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)* (IEEE, 2015), pp. 532–533.
68. Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **13**, 600–612 (2004).
69. T. A. Driscoll, L. N. Trefethen, *Schwarz-Christoffel Mapping* (Cambridge Univ. Press, 2002).
70. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.* **5**, eabc5986 (2020).

Acknowledgments: We would like to thank C. Feng for insights on network structure design and the discussion on environmental and path representations. His feedback provided us with valuable assistance. We express our gratitude to T. Wu for invaluable suggestions on the manuscript. Our heartfelt thanks go out to L. Xu, R. Zhang, G. Xu, M. Wang, M. Zhang, and other fellow students from the FASTLAB who assisted during outdoor experiments. **Funding:** This work was supported by the National Natural Science Foundation of China under grant no. 62322314, the National Key R&D Program of China under grant no. 2023YFB4706600, and HUAWEI Application Innovation Lab. **Author contributions:** Z.H. participated in the design and implementation of the front end and back end, contributed to manuscript writing, and conducted experiments. M.T. participated in the design and implementation of the front end, designed the maze experiment and fixed-wing navigation, and contributed in the partial manuscript writing. Z.G. and D.X. were involved in the maze experiment and provided consultation. J.X. fine-tuned the fixed-wing experiments. Q.W. was involved in developing the localization module for the maze experiment and offered input. Y.G. was engaged in graphic design and lent his guidance. J.W. designed the visualization of the benchmark in the front end. C.X. provided people and sites for experimental testing. F.G. supervised the research, provided financial support, offered crucial advice on software and system debugging, and revised the manuscript. **Competing interests:** The authors declare that they have no competing interests. **Data and materials availability:** The data and source code in the paper are available at https://doi.org/10.5281/zenodo.14288814 and https://github.com/ZJU-FAST-Lab/DPtraj.

Submitted 15 August 2024
Accepted 23 May 2025
Published 18 June 2025
10.1126/scirobotics.ads4551

Science Robotics

Hierarchically depicting vehicle trajectory with stability in complex environments

Zhichao Han, Mengze Tian, Zaitian Gongye, Donglai Xue, Jiaxi Xing, Qianhao Wang, Yuman Gao, Jingping Wang, Chao Xu, and Fei Gao

Sci. Robot. **10** (103), eads4551. DOI: 10.1126/scirobotics.ads4551

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.