Technical Report: Learning to Plan Maneuverable and Agile Flight Trajectory with Optimization Embedded Networks

Zhichao Han[*,1,2], Long Xu[*,1,2] and Fei Gao[1,2]

∗ Equal contribution
[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: Fei Gao Email: fgaoaa@zju.edu.cn

Abstract

In recent times, an increasing number of researchers have been devoted to utilizing deep neural networks for end-to-end flight navigation. This approach has gained traction due to its ability to bridge the gap between perception and planning that exists in traditional methods, thereby eliminating delays between modules. However, the practice of replacing original modules with neural networks in a black-box manner diminishes the overall system's robustness and stability. It lacks principled explanations and often fails to consistently generate high-quality motion trajectories. Furthermore, such methods often struggle to rigorously account for the robot's kinematic constraints, resulting in the generation of trajectories that cannot be executed satisfactorily. In this work, we combine the advantages of traditional methods and neural networks

by proposing an optimization-embedded neural network. This network can learn high-quality trajectories directly from visual inputs without the need of mapping, while ensuring dynamic feasibility. Here, the deep neural network is employed to directly extract environment safety regions from depth images. Subsequently, we employ a model-based approach to represent these regions as safety constraints in trajectory optimization. Leveraging the availability of highly efficient optimization algorithms, our method robustly converges to feasible and optimal solutions that satisfy various user-defined constraints. Moreover, we differentiate the optimization process, allowing it to be trained as a layer within the neural network. This approach facilitates the direct interaction between perception and planning, enabling the network to focus more on the spatial regions where optimal solutions exist. As a result, it further enhances the quality and stability of the generated trajectories.

Introduction

Unmanned aerial vehicles (UAVs) have gained widespread adoption in various societal domains, such as aerial photography, exploration, and search and rescue, due to their compact hardware design and agile maneuverability. Efficient and robust navigation modules play a crucial role in achieving UAV autonomy, attracting significant attention from both academia and industry.

Traditional navigation tasks can be divided into perception and motion planning. Perception modules process raw data from sensors such as depth cameras to construct an occupancy map and derive environment representations conducive to motion planning, such as Euclidean signed distance fields (ESDF) (*1*) or neural radiance fields (NeRF) (*2*). Motion planning modules, on the other hand, utilize the constructed map along with robot state, kinematics, and

obstacle avoidance constraints to compute energy-minimizing trajectories with time regularization. While traditional navigation strategies offer intuitive engineering solutions and theoretical completeness and interpretability, they suffer from delays in the modular decomposition framework, adversely affecting agile flight capabilities of quadcopters (*3*). Moreover, the layered framework can cause a lack of interconnections between sub-modules, making the flight performance susceptible to sensor noise and often requiring manual fine-tuning of numerous parameters by engineers (*4*).

In recent years, there has been increasing research interest in directly fusing perception and planning modules into a single neural network. This end-to-end pipeline enables learning motion directly from sensor data, bypassing explicit mapping. However, such strategies result in a black-box navigation system, making debugging challenging. Furthermore, attaining a favorable balance between dynamic feasibility, obstacle avoidance, and high-quality trajectory generation places a significant burden on the network. For instance, due to physical platform or task constraints, it is often desired to impose dynamic constraints such as maximum velocity and acceleration constraints on the trajectories. To address these issues, researchers often resort to designing complex strategies for the network, which can impact optimality and still fail to guarantee dynamic feasibility of the trajectories.

In this work, we combine the strengths of traditional trajectory optimization and neural networks to develop an end-to-end visual navigation system capable of generating trajectories directly from depth information without the need for explicit mapping. A key feature of our approach, compared to conventional learning-based motion planning algorithms, is the incorporation of numerical optimization within the neural network, coupled with joint training. This method alleviates the burden on the network, enhances the interpretability of the system, and ensures optimal and dynamically feasible motion trajectories. Moreover, our method is scalable and allows for the inclusion of additional user-defined constraints without the need

for retraining. In our technical approach, we utilize the neural network to directly extract safe guidance regions from the depth information, which are then transformed into geometric spatial constraints considered during trajectory optimization. The trajectory optimization process incorporates these constraints as safety boundaries while simultaneously integrating user-specified dynamic constraints, resulting in efficient convergence ($\sim$1ms) to high-quality trajectories. This ensures that the quadcopter remains maneuverable and agile even in complex environments. Unlike conventional learning-based black-box navigation systems, our optimization algorithm, enabled by a clear mathematical model, robustly converges to optimal solutions within the feasible topological space generated by the network. Furthermore, by making numerical optimization differentiable, it can be modeled as a layer and trained jointly with the neural network. This allows the gradient of the evaluation loss of the trajectories to be directly backpropagated to the network, enabling the network to focus on the spatial regions where the optimal trajectories reside. Moreover, to ensure sufficient exploration of the environment during actual flight, we introduce motion primitives within the network. The network outputs the selection probabilities for each motion primitive, and based on these probabilities, safe and feasible spaces are assigned to certain primitives. In practical applications, we can parallelly perform trajectory optimization within the safe spaces represented by high-probability motion primitives, and select the optimal trajectory as the execution plan. The main contributions of this paper can be summarized as follows:

- We have designed a lightweight neural network capable of directly identifying valuable motion primitives from depth information, which are then used to generate the necessary safety spatial constraints for subsequent numerical optimization.

- By making trajectory optimization differentiable, we treat it as a layer and train it jointly with the network. This seamless integration enables the neural network to evolve directly

towards optimal trajectories, eliminating any gaps between the optimization and learning processes.

- By leveraging the strengths of neural networks and numerical optimization, we propose a high-quality, map-free planning approach. This approach enables the instantaneous generation of optimal and safe trajectories, strictly adhering to dynamic constraints.

Related Work

Classical Motion Planning Algorithms

Gradient-based motion planning (*5–10*) is widely adopted for generating local trajectories for UAVs, treating the problem as constrained nonlinear optimization. Such methods typically require the explicit construction of the environment through depth information, followed by the manual design of strategies to extract safety constraints for trajectory optimization. Euclidean Signed Distance Fields (ESDF) are widely used for modeling safety constraints, as they provide signed distance and gradient information from any grid point to obstacles within the space (*5, 6*). However, constructing ESDF incurs additional computational costs and involves a trade-off between efficiency and accuracy, as higher resolutions exponentially increase computation and memory requirements. Zhou et al. (*7*) proposed the well-known ego planner, which avoids ESDF construction by continuously generating safe guidance paths within an iterative framework to provide obstacle avoidance gradients. However, this method lacks convergence guarantees and may be prone to getting trapped in unsafe local minima, especially in complex environments. Furthermore, using guidance paths to deform trajectories deviates from the original trajectory optimization problem formulation, affecting optimality. Corridor-based methods (*9, 10*) have also gained popularity in the field of local motion planning. These methods extract feasible convex hulls from the environment point cloud using geometric computations

to model safety constraints as linear or cone constraints. However, these methods require an additional collision-free path to provide seed points for the convex hull. Such paths are often obtained using low-dimensional search algorithms like A* or hybrid A*. These search algorithms often do not consider the robot's higher-order kinematics, resulting in convex hulls that are not conducive to generating maneuverable and dynamically feasible trajectories.

Learning-Based Motion Planning Algorithms

Learning-based methods (*3, 4, 11–16*) have emerged as promising approaches in the field of local planning, eliminating the need for explicit mapping and reducing latency. Loquercio et al. (*3*) leveraged deep convolutional neural networks to learn flight trajectories from depth images, using human pilot trajectories as supervision. However, this method requires high-quality and large-scale datasets. Yang et al. (*13*) proposed Iplanner, which incorporated safety loss in the trajectory generation process to emphasize obstacle avoidance. Their work further improved by fusing semantic information (*14*), allowing the robot to consider terrain features, which is crucial for quadrupedal navigation. Kulkarni et al. (*15*) employed reinforcement learning for end-to-end navigation and enhanced safety through a custom depth collision encoder. These methods heavily rely on the capabilities of neural networks and lack principled guarantees regarding kinematic feasibility and trajectory optimality. Recently, some approaches have combined networks with numerical optimization. For instance, a particular work (*16*) learned collision probabilities for any point in space using a network, which were further modeled as safety constraints in trajectory optimization. Similarly, another work (*4*) addressed finer obstacle avoidance by modeling the robot's shape as a convex hull and predicting the signed distance between the hull and the nearest obstacle using a neural network. Although these works integrate networks and optimization for robot navigation, a significant difference compared to our approach is that the network and optimization are independent components in the aforemen-

tioned works. In contrast, our algorithm incorporates differentiable optimization as part of the network training process. Consequently, our method facilitates the evolution of the network towards directions beneficial for subsequent optimization, ultimately generating higher-quality solutions while ensuring maneuverability and agile flight.

Methodology

End-to-End Navigation System Overview

In this work, we employ a unique class of trajectories called $MINCO$ (*17*) to represent flight trajectories ξ. $MINCO$ is a special multi-piece polynomial representation parameterized by piece durations $\mathbf{T} = [T_1, T_2, ..., T_N]^T \in \mathbb{N}+$ and waypoints $\mathbf{q} = [q_1, q_2, ..., q_{N-1}] \in \mathbb{R}^{3\times(N-1)}$, where N is denoted as the number of trajectory pieces. This compact representation naturally satisfies the state constraints at the start and end points, as well as the high-order continuity of adjacent polynomials at the waypoints, without the need for additional constraints. Building upon this representation, the trajectory optimization is formulated as the minimization of control energy with first-order temporal regularization, and can be expressed as follows:

$$\min_{\mathbf{q},\mathbf{T}} J = \int_0^{||\mathbf{T}||_1} (\xi^{(u)}(t))^{\mathrm{T}} \mathrm{W} \xi^{(u)}(t) dt + \rho ||\mathbf{T}||_1 \tag{1}$$

$$s.t. \quad \mathcal{G}(\xi(t), \xi^{(1)}(t), ..., \xi^{(u)}(t)) \leq 0, \forall t \in [0, ||\mathbf{T}||_1] \tag{2}$$

$$\xi(t) \in \mathcal{F}, \forall t \in [0, ||\mathbf{T}||_1], \tag{3}$$

where W is a positive definite energy weight matrix. ρ is the temporal regularization weight and u represents the dimension of the control variable. $\mathcal{G}$ represents pre-defined kinematic constraints, which are specifically formulated based on user's requirements and the robot's dynamics. Additionally, Eq. (3) represents the obstacle avoidance constraints based on the flight corridor $\mathcal{F}$. Typically, this constraint can be accurately modeled as a linear or conic constraint with respect to the robot's position coordinates. Due to advancements in the field of opti-

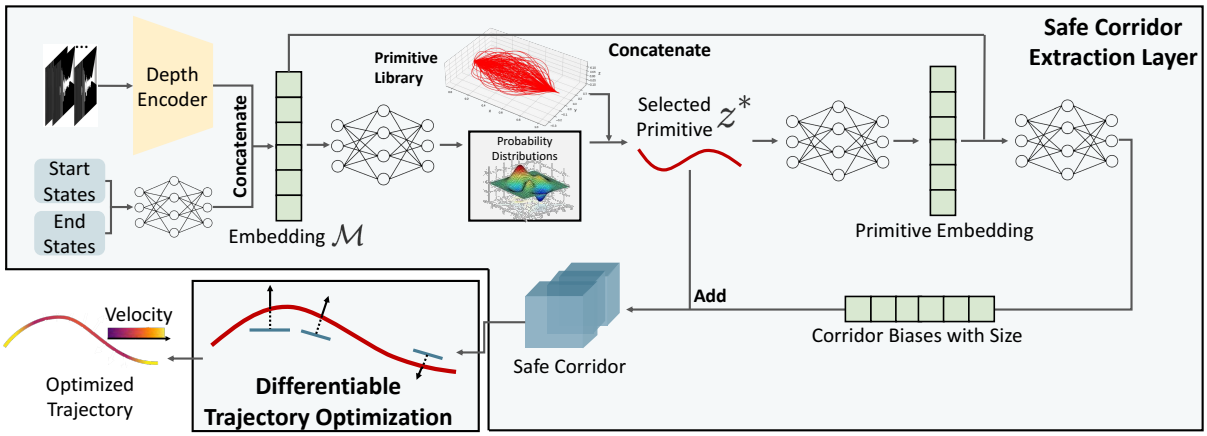

Fig. 1. Navigation framework.

mization, once the trajectory optimization Eq. (1-3) is fully formulated and modeled, mature gradient-based numerical methods are available that can efficiently converge to high-quality solutions at low computational cost. However, accurately extracting feasible spaces from complex and cluttered environments during actual flight poses a significant challenge. As mentioned in Sect. I, it involves multiple modules such as sensor data processing, mapping, graph search, etc., and is susceptible to noise, making this aspect the Achilles' heel of the entire navigation system. Therefore, in this work, we intuitively explore the utilization of neural networks to learn flight corridors directly from depth images, without the need for explicit mapping. Essentially, we employ neural networks to learn safety constraints within the trajectory optimization process.

Our navigation framework, as illustrated in Fig. 1, can be broadly divided into a network layer for extracting flight corridors and a differentiable trajectory optimization layer. The network layer takes inputs such as depth measurements, the current state of the robot, and the target point's position to output flight corridors. Subsequently, the trajectory optimization layer plans a high-quality spatial-temporal optimal trajectory constrained within the flight corridors, while strictly adhering to specified dynamic constraints. Furthermore, the inputs and outputs of

each module, such as the target point, flight corridors, and planned trajectory, are normalized with respect to the robot's current body frame. This normalization enhances the generalization capability of the model and reduces the dependence of the entire navigation system on global localization. In the following sections, we will first introduce the structural design of the network layer and provide a detailed description of its output. Then, we discuss the gradient propagation of the optimal solution generated by the trajectory optimization layer with respect to spatial constraints, which forms the foundational basis for jointly training the embedded optimization within the network.

Learning-Based Safe Corridor Extraction Layer

Before delving into the specific details of the network structure, we first instantiate its output representation. For continuous-time constraints Eq. (3), similar to the approach (*18*), we discretize each piece of the polynomial into λ constraint points. Subsequently, we control the entire trajectory by imposing constraints at these constraint points:

$$\begin{aligned} \xi(t) \in \mathcal{F}, \forall t \in [0, ||\mathbf{T}||_1] \iff \\ \xi_i(\frac{j}{\lambda T_i}) \in \mathcal{F}_{i,j}^{\phi} \forall i \in [1, ..., N], \forall j \in [1, ..., \lambda], \end{aligned} \tag{4}$$

where ϕ is the parameters of the neural network. The physical significance of Eq. (4) lies in the fact that the neural network needs to assign a safety convex hull to each constraint point along the trajectory. Consequently, the network is required to output a total of $N\lambda$ convex hulls. Moreover, we would like to emphasize that the network has the theoretical capability to output convex hulls of arbitrary shapes. However, for the sake of simplicity and ease of understanding, we define each convex hull as a cube parameterized by its center and length.

Here, we design a novel neural network based on motion primitives, which enables us to fulfill the aforementioned requirements. This network initially employs deep convolutional and

fully connected layers to extract features from depth images and the initial and final states of the robot. These features are fused to obtain a latent representation, denoted as $\mathcal{M}$. Subsequently, $\mathcal{M}$ is further processed through a neural network to output a probability distribution over a pre-built library of motion primitives ζ. It is worth mentioning that each motion primitive z, in order to align with the subsequent corridor parameters, is represented by $N\lambda$ points. Furthermore, the selected motion primitive z^* and $\mathcal{M}$ are jointly fed into the final corridor generation module. One of its roles is to refine the motion primitives for improved accuracy, with the modified point coordinates serving as the centers of the safety cubes. Additionally, this module is responsible for assigning the corresponding length to each cube. It is worth noting that the advantages of this motion-primitive-based structure are evident in at least two aspects. Firstly, compared to directly regressing the final cube centers, our approach first obtains a probability distribution and selects better motion primitives. This process can be modeled as a classification problem, which reduces the network's burden and facilitates learning and convergence. Secondly, during practical deployment, we have the flexibility to select multiple motion primitives based on their probabilities. This allows for parallel optimization of multiple trajectories, thereby enhancing the robot's exploration capabilities in diverse environmental topologies. Simultaneously, it also increases the system's fault tolerance.

Regarding the construction of the motion primitive library, in our practical experiments, we recorded tens of thousands of trajectories using classical navigation algorithms. These trajectories are uniformly discretized into $N\lambda$ points and transformed into the local body coordinate system. To eliminate unnecessary duplicate motion primitives and limit the size of the library, we normalize the endpoint distances and directions for all motion primitive data. Finally, we employ the k-means algorithm to cluster the processed dataset and collect approximately 100 elite motion primitives as the library.

Differentiable Numerical Optimization Layer

In this section, we discuss how to make the optimization process differentiable, allowing us to backpropagate the gradients of the loss applied to the trajectory onto the network parameters during training. For simplicity, we denote the optimization variables as $x = (\mathbf{q}, \mathbf{T})$. Then, the nonlinear optimization problem Eq. (1-3) with inequality constraints can be generally reformulated as follows:

$$\min_{x} J = J(x) \tag{5}$$

$$s.t. \quad F(x, \phi) \leq 0. \tag{6}$$

Here, F represents a general formulation that encompasses the original constraints Eq. (2,3), and ϕ denotes the neural network parameters. Assuming x^* is the optimal solution to this optimization problem, and $\mathcal{L}$ is the evaluation loss applied to the trajectory during training, the gradient of the neural network can be computed as follows:

$$\nabla\phi\mathcal{L} = \nabla_{\phi}x\nabla_{x}\mathcal{L} \tag{7}$$

Generally, the term $\nabla_{x}\mathcal{L}$ can be analytically computed. Therefore, our focus now shifts to discussing the estimation of $\nabla_{\phi}x$. Due to the usage of gradient-based numerical solvers, a intuitive method for estimating parameter gradients, known as unrolling (*19–23*), involves maintaining the entire computational graph throughout the iteration process. However, this approach poses significant challenges in terms of memory usage and efficiency, particularly when dealing with complex problem formulations. Moreover, it may also face issues related to gradient divergence or vanishment. In this work, we assume the efficient attainment of the optimal solution x^* to the problem, and thus employ the implicit function differentiation theorem from Dontchev and Rockafellar (*24*). This method relies on leveraging the first-order optimality condition (*25, 26*) of the optimization problem to analytically estimate the gradients of the parameters without

the need for explicit unrolling of the entire iteration process. The Lagrangian function for this optimization problem is as follows:

$$L(x, \lambda) = J(x) + \lambda^T F(x, \phi). \tag{8}$$

Then, the corresponding KKT (Karush-Kuhn-Tucker) conditions are as follows:

$$\begin{aligned} \nabla_x J + \nabla_x F \lambda^* &= 0, \\ D(\lambda^*) F(x^*, \phi) &= 0, \\ F(x^*, \phi) &\leq 0, \\ \lambda^* &\geq 0, \end{aligned} \tag{9}$$

where $D(\cdot)$ denotes a diagonal matrix from a vector. Then, we apply the total differential operator d to the equations in KKT conditions:

$$\begin{aligned} (\nabla_{x,x} J + (\nabla_{x,x} F \lambda^*)^{\mathrm{T}}) \mathrm{d}x + \nabla_x F \mathrm{d}\lambda + (\nabla_{x,\phi} F \lambda^*)^{\mathrm{T}} \mathrm{d}\phi &= 0, \\ D(F)\mathrm{d}\lambda + D(\lambda^*) \nabla_x F \mathrm{d}x + D(\lambda^*) \nabla_\phi F \mathrm{d}\phi &= 0. \end{aligned} \tag{10}$$

Subsequently, Eq. (10) is further transformed into a compact matrix form:

$$\begin{bmatrix} \mathrm{d}x \\ \mathrm{d}\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_{x,x} J + (\nabla_{x,x} F \lambda^*)^{\mathrm{T}}) & \nabla_x F \\ D(\lambda^*) \nabla_x F & D(F) \end{bmatrix}^{-1} \begin{bmatrix} (\nabla_{x,\phi} F \lambda^*)^{\mathrm{T}} \\ D(\lambda^*) \nabla_\phi F \end{bmatrix} \mathrm{d}\phi. \tag{11}$$

By solving this system of equations, we can analytically obtain the desired Jacobian matrix $\nabla_\phi x$, which in turn allows us to derive the final parameter gradients $\nabla \phi \mathcal{L}$.

Results

Ablation Experiments

To verify the effectiveness of embedding the numerical optimization into the neural network, we conducted ablation experiments in a test set with more than ten thousand items of data,

quantitatively comparing the safety of guidance regions extracted by the neural network and the energy consumption of the optimized trajectories, as shown in Table 1. We consider it unsafe when the edge of the guidance region touches any obstacle. Besides, the integral with respect to time of the square of the jerk of the trajectory is utilized to measure the energy consumption.

Table 1: Ablation Experiments.

	Safety Ratio ↑	Avg. Energy ↓
w/o Opt.	82.1%	7.262
w/ Opt.	85.2%	5.285

Since we remove the imitation of the ground truth and add the loss function minimizing the energy of the trajectory after embedding numerical optimization in the neural network, the safe regions extracted by the network are quickly tuned towards the direction of reducing energy consumption. At the same time, as mentioned in the previous section, with backpropagation of the energy loss function, we can remove the requirement for the safe region to be as large as possible, which makes the security constraints of the region easier to satisfy, allowing the network to focus more on containing the optimal solution, thus resulting in a higher safety ratio.

Simulation Experiments

We deploy the algorithm on a simulated quadcopter, conducting experiments in an environment with dynamics simulation, as shown in Fig. 2. The size of the simulation environment is 50m $\times$ 50m, which contains randomly generated obstacles in the form of columns and loops. With known localization, we require the quadcopter to start from a random location at the edge of the map and continuously replan to traverse the entire unknown environment.

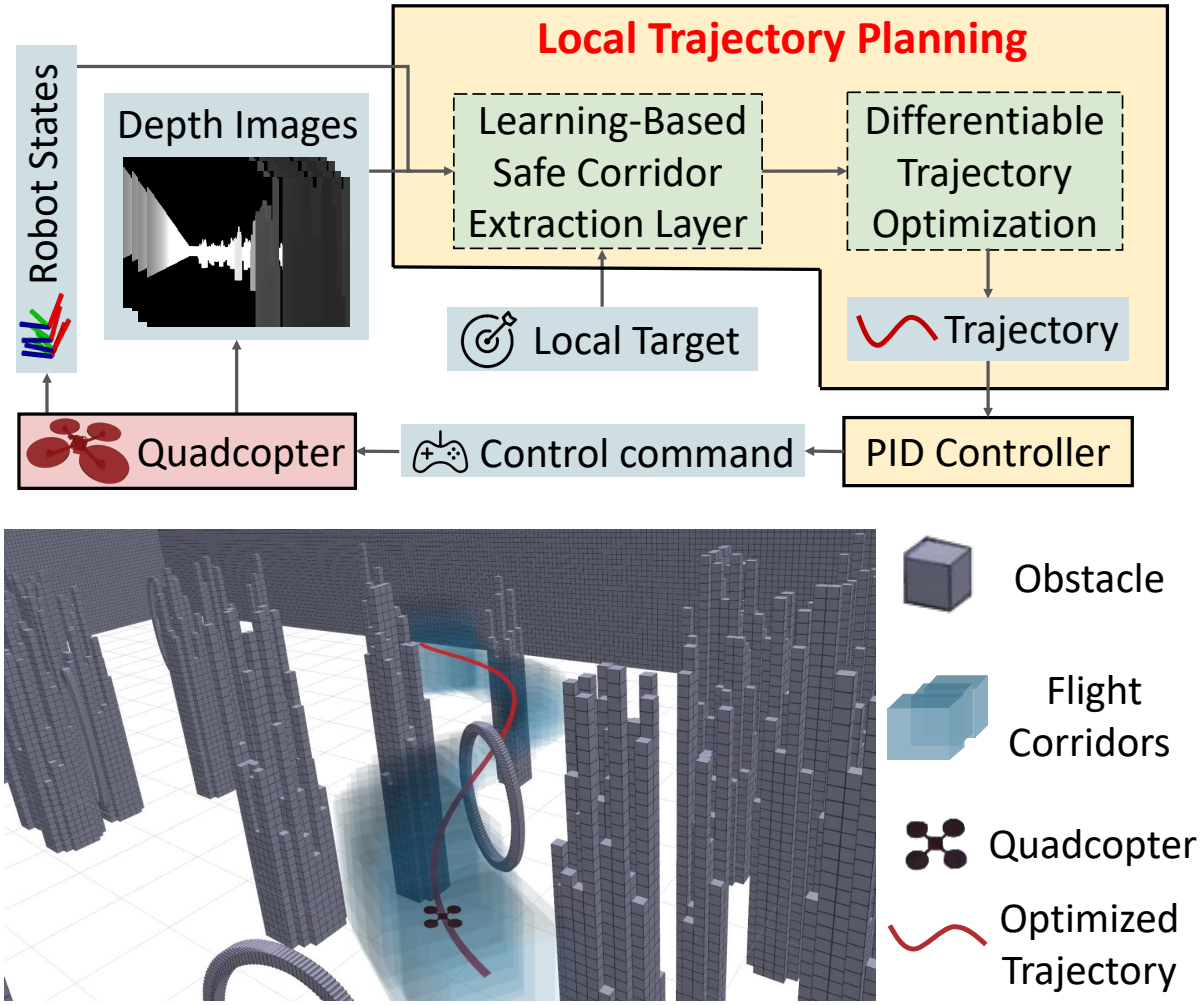

Fig. 2. Simulation experiments and system framework.

In addition, we quantitatively compare the proposed method with the state-of-the-art algorithm Ego-planner (*7*) in terms of objective function value and total processing latency, with more than four thousand comparative tests. All simulations are run on a desktop with an Intel i9-10900K CPU and a Nvidia GeForce RTX3070Ti GPU.

In Table 2, our approach achieves better performance than Ego-planner (*7*). The use of collision-free trajectories as ground truth to supervise and training based on backpropagation of the results of trajectory optimization make the neural network give safety regions that more

Table 2: Algorithm Comparisons.

	Objective Fu -nction Value ↓	Total Proces- sing Latency ↓
proposed	59.90 ± 8.71	3.49 ± 0.73ms
Ego-planner (*7*)	61.65 ± 11.73	24.33 ± 16.58ms

easily contains the optimal solution. Thus, the proposed algorithm has a greater advantage in terms of objective function value. Compared to classical pipelines, neural networks have more stable inference times and outputs, bringing low standard deviations. Also, the end-to-end pipeline and parallel GPU-based reasoning make it have lower latency.

Discussion

In this work, we propose a novel planning framework for maneuverable and agile flight of quadrotors. Using an embedded optimized neural network, we plan trajectories directly from visual measurements without the need for explicit mapping, while being able to guarantee the dynamical feasibility of the trajectories. Benefiting from the differentiable trajectory optimization, the burden on the neural network is relieved to more easily extraction of safe regions containing optimal solutions. Simulation experiments prove the efficiency of the pipeline compared to state-of-the-art method. In the future, more quantitative experiments will be carried out in high speed flights in complex environments, where the success rate of classical methods will plummet due to high latency. We will also further optimize the proposed pipeline and conduct more comparisons with other learning-based methods. Besides, this framework will be deployed to real quadrotors for more tests.

References

1. L. Han, F. Gao, B. Zhou, S. Shen, Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 4423–4430.

2. M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, M. Schwager, Vision-only robot navigation in a neural radiance world, *IEEE Robotics and Automation Letters* **7**, 4606–4613 (2022).

3. A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, D. Scaramuzza, Learning high-speed flight in the wild, *Science Robotics* **6**, eabg5810 (2021).

4. R. Han, S. Wang, S. Wang, Z. Zhang, J. Chen, S. Lin, C. Li, C. Xu, Y. C. Eldar, Q. Hao, *et al.*, Neupan: Direct point robot navigation with end-to-end model-based learning, *arXiv preprint arXiv:2403.06828* (2024).

5. B. Zhou, J. Pan, F. Gao, S. Shen, Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight, *IEEE Transactions on Robotics* **37**, 1992–2009 (2021).

6. B. Zhou, F. Gao, L. Wang, C. Liu, S. Shen, Robust and efficient quadrotor trajectory generation for fast autonomous flight, *IEEE Robotics and Automation Letters* **4**, 3529–3536 (2019).

7. X. Zhou, Z. Wang, H. Ye, C. Xu, F. Gao, Ego-planner: An esdf-free gradient-based local planner for quadrotors, *IEEE Robotics and Automation Letters* **6**, 478–485 (2020).

8. B. Zhou, F. Gao, J. Pan, S. Shen, Robust real-time uav replanning using guided gradient-based optimization and topological paths, *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020), pp. 1208–1214.

9. F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, S. Shen, Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments, *IEEE Transactions on Robotics* **36**, 1526–1545 (2020).

10. J. Tordesillas, J. P. How, Mader: Trajectory planner in multiagent and dynamic environments, *IEEE Transactions on Robotics* **38**, 463–476 (2021).

11. R. E. Allen, M. Pavone, A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance, *Robotics and Autonomous Systems* **115**, 174–193 (2019).

12. G. Chou, D. Berenson, N. Ozay, Uncertainty-aware constraint learning for adaptive safe motion planning from demonstrations, *Conference on Robot Learning* (PMLR, 2021), pp. 1612–1639.

13. F. Yang, C. Wang, C. Cadena, M. Hutter, iplanner: Imperative path planning, *arXiv preprint arXiv:2302.11434* (2023).

14. P. Roth, J. Nubert, F. Yang, M. Mittal, M. Hutter, Viplanner: Visual semantic imperative learning for local navigation, *arXiv preprint arXiv:2310.00982* (2023).

15. M. Kulkarni, K. Alexis, Reinforcement learning for collision-free flight exploiting deep collision encoding, *arXiv preprint arXiv:2402.03947* (2024).

16. M. Jacquet, K. Alexis, N-mpc for deep neural network-based collision avoidance exploiting depth images, *arXiv preprint arXiv:2402.13038* (2024).

17. Z. Wang, X. Zhou, C. Xu, F. Gao, Geometrically constrained trajectory optimization for multicopters, *IEEE Transactions on Robotics* **38**, 3259–3278 (2022).

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

19. C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, *International conference on machine learning* (PMLR, 2017), pp. 1126–1135.

20. M. Bhardwaj, B. Boots, M. Mukadam, Differentiable gaussian process motion planning, *2020 IEEE international conference on robotics and automation (ICRA)* (IEEE, 2020), pp. 10598–10604.

21. B. A. Pearlmutter, J. M. Siskind, Reverse-mode ad in a functional framework: Lambda the ultimate backpropagator, *ACM Transactions on Programming Languages and Systems (TOPLAS)* **30**, 1–36 (2008).

22. C. Zhang, V. Lesser, Multi-agent learning with policy prediction, *Proceedings of the AAAI Conference on Artificial Intelligence* (2010), vol. 24, pp. 927–934.

23. T. Han, Y. Lu, S.-C. Zhu, Y. N. Wu, Alternating back-propagation for generator network, *Proceedings of the AAAI Conference on Artificial Intelligence* (2017), vol. 31.

24. A. L. Dontchev, R. T. Rockafellar, *Implicit functions and solution mappings*, vol. 543 (Springer, 2009).

25. B. Amos, J. Z. Kolter, Optnet: Differentiable optimization as a layer in neural networks, *International Conference on Machine Learning* (PMLR, 2017), pp. 136–145.

26. A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, J. Z. Kolter, Differentiable convex optimization layers, *Advances in neural information processing systems* **32** (2019).