IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

# LPNet: A Reaction-based Local Planner for Autonomous Collision Avoidance using Imitation Learning

Junjie Lu, Bailing Tian, Hongming Shen, Xuewei Zhang, and Yulin Hui

***Abstract*—In this work, we propose a reaction-based local planner for autonomous collision avoidance of quadrotor in obstacle-cluttered environment without relying on an explicit map. Our approach searches for feasible trajectory using a set of motion primitives in state lattice and represents the optimal one as a polynomial by solving an optimal control problem. A modified Q-network, termed LPNet, is presented to predict the action-values of motion primitives from the current depth image and the state estimation of the quadrotor directly. To train the proposed LPNet, a primitive-based expert policy with privileged information about the surroundings and unconstrained computational budget is developed to provide demonstrations for imitation learning. Finally, a series of experiments are conducted to demonstrate the effectiveness and time-efficiency of the proposed method in both simulation and real-world.**

***Index Terms*—Integrated Planning and Learning, Collision Avoidance, Aerial Systems: Perception and Autonomy**

## I. Introduction

UNMANNED aerial vehicles (UAVs) have been involved in many applications in recent years, such as exploration, photography, and drone racing. To ensure the safe, smooth, and swift flight in obstacle-dense environment, an effective planner is essential for trajectory generation relying only on lightweight airborne sensors and limited computational resources.

The navigation problem of quadrotors has been extensively studied. Traditionally, most of the presented works separate it into localization, mapping, and trajectory optimization. Localization and mapping are utilized to integrate sensor measurements into a map and update the distance information against obstacles continuously for trajectory optimization. Subsequently, the smooth and collision-free trajectories are generated based on the incrementally constructed map. The divide-and-conquer pipelines facilitate the independent optimization of individual modules and make the overall system more interpretable. However, the separation strategy introduces additional latency and is sensitive to the sensor noise, making it more challenging for autonomous flight in obstacle-dense environment.

Manuscript received: April 13, 2023; Revised July 21, 2023; Accepted August 29, 2023. This paper was recommended for publication by Editor Hanna Kurniawati upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the National Natural Science Foundation of China under Grants 62273249, 62203415, 62022060, and 62073234 (Corresponding author: Bailing Tian).

The authors are with the School of Electrical and Information Egineering, Tianjin University, Tianjin, 300072, China. {lqzx1998, bailing_tian, shenhm, zhangxuewei, huiyulin}@tju.edu.cn

Digital Object Identifier 10.1109/LRA.2023.3314350

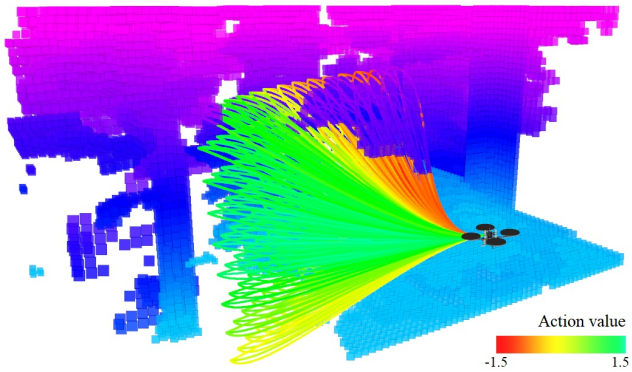



Fig. 1. Visualization of motion primitives and the action-values predicted by the proposed LPNet. Note that the map is unavailable and only the optimal trajectory will be solved in practice. More details can be found in the attached video at https://youtu.be/4PaDz5MERtE.

With recent breakthroughs in deep learning, the learning-based planners which process raw sensory observations and perform complex behaviors directly have shown great potential and gain more and more popularity. Some policies are trained to imitate a privileged expert or experienced pilot, and others adopted reinforcement learning to explore the optimal policy by trial-and-error in simulation. Neural networks have a unique ability to discover patterns from imprecise or abstract data which is difficult to model mathematically, such as image, audio, and text. However, for quadrotors with explicit dynamic and kinematic model, the end-to-end policy wastes a lot of effort to approximate the kinodynamic model and is challenging to be deployed in real-world.

In this work, a reaction-based local planner is proposed to give quick reactions to sensor observations and generate smooth, dynamically feasible, and collision-free trajectories without relying on an explicit map. Given the current state estimation of UAV, our approach explores the space using a set of motion primitives in state lattice. A modified Q-network, termed the LPNet, is adopted to predict the action-values (also known as $Q$ in the field of reinforcement learning) of motion primitives in parallel from the current depth image and the state estimation directly. Since the depth image reflects the free-space directly, the foundation models such as ResNet-50 are superfluous. Therefore, we modified a lightweight backbone for depth feature extraction. Compared to the direct association between inputs and targets in supervised learning, the reward signal of reinforcement learning algorithm is frequently sparse, noisy and delayed, making it harder to

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

converge. To this end, a primitive-based expert policy which has access to the privileged information about the surrounding and unconstrained computational budget is proposed to serve as the optimal action-value function $Q^*(s^*, a)$ for supervision, where $s^*$ is the privileged observations and $a$ is the predefined motion primitives. Besides, the primitives are represented as polynomial trajectories by solving the optimal control problem, which ensures the feasibility of trajectories and allows our network to concentrate on action-value inference. At the test, all action-values are predicted in one propagation, and only the optimal motion primitive is solved and executed in a receding horizon fashion. That is, the computational cost of the learning-based policy is slightly affected by the number of primitives while increasing linearly in the expert policy.

In contrast to traditional algorithms, the presented method does not rely on explicit map and is able to generate smooth, safe, and dynamically feasible trajectories from noisy sensory observations directly in a much shorter time, which is essential for the safety of autonomous flight in obstacle-dense environment. The main contributions are summarized as follows:

1) An efficient learning-based local planner is proposed, which directly maps the noisy sensory observations to smooth, collision-free, and dynamically feasible trajectories without mapping.
2) A lightweight network, termed LPNet, is presented to predict the action-values of motion primitives in state lattice space and the optimal one is represented as a polynomial trajectory by solving an optimal control problem.
3) The LPNet is trained by imitating a primitive-based expert policy, which has access to the privileged information about the surroundings and unconstrained computational budget.

## II. Related Work

The problem of autonomous navigation has been extensively studied and is usually divided into mapping and planning subtasks. One of the earliest work [1] proposes the method of minimum-snap trajectory generation, where the trajectories are represented as piecewise polynomials and generated by solving a quadratic programming problem. Furthermore, [2] ensures the safety of the trajectories by iteratively inserting intermediate waypoints to the colliding segments. [3], [4] adopt the idea of safe-flying corridor, in which the trajectories are constrained into an obstacle-free space consisting of multiple convex shapes. By contrast, the soft-constrained methods optimize trajectories by minimizing the cost function considering the smoothness, safety, and dynamic penalties simultaneously. Specifically, [5], [6] address the trajectory optimization problem by gradient descent, while [7], [8] approximate the optimal trajectory using forward sampling of stochastic diffusion processes. In addition, some methods [9], [10] search the feasible trajectories from a pre-designed library of motion primitives, which is instructive for the proposed method. Among the above approaches, the map represented by point clouds, octrees, or occupancy grid is crucial for safe-flying corridor generation, and some works [11] are dedicated to Euclidean Signed Distance Field (ESDF) computation for gradient-based optimization. The divide-and-conquer pipelines facilitate the independent optimization of individual modules but introduce additional latency, making it more challenging for aggressive flight in obstacle-dense environment.

In contrast to these approaches, the breakthroughs of deep learning inspire alternative solutions to autonomous navigation without explicit mapping and planning stages. Some works adopt reinforcement learning to explore the optimal policy by trial-and-error for collision avoidance [12], gap traversing [13], and drone racing [14]. For data efficiency, the privileged expert, usually a traditional navigation approach, is adopted to give demonstrations and imitated by the student policy in [15], [16]. The paradigm of imitation learning combines the performance of state-of-the-art planning methods with the perceptual awareness of neural networks. Additionally, [17], [18] train the learning-based policy to predict the collision probabilities from a pre-collected dataset by supervised learning. However, some of them [12], [17] execute the predicted control commands (velocity or acceleration, for example) directly and neglect the kinematic smoothness of trajectory, making it challenging for real-world deployment. Whereas other methods [13], [14], [16] treat the surroundings as prior and concentrate on specific tasks (control or planning) without considering perception. Inspired by above methods, we present a learning-based local planner that predicts the action-values of motion primitives by a lightweight network and generates the kinodynamic feasible trajectory from sensor measurements without mapping.

## III. Method

Let $\mathcal{A}$ denotes the predefined primitive library with fixed final positions and forward speeds sampled within the FOV of the depth sensor. The proposed LPNet is designed to predict the action-values $Q(s, a)$ of all primitives, where $a = (\mathbf{p}, \mathbf{v}) \in \mathcal{A}$ represents the primitive and $s$ is the sensor observations including current velocity $\mathbf{v}_0$, acceleration $\mathbf{a}_0$, and depth image $\mathcal{D}$. Subsequently, let $\mathbf{x}(t) = [\mathbf{p}(t)^T, \dot{\mathbf{p}}(t)^T, \ddot{\mathbf{p}}(t)^T]^T$ be the state vector of quadrotor and $\mathbf{u}(t) = \dddot{\mathbf{p}}(t)$ denote the control input in differentially flat space. The optimal primitive is 1-D time-parametrized as a polynomial trajectory $\mathbf{p}(t) = [p_x(t), p_y(t), p_z(t)]^T$ specified independently in each axis by solving an optimal control problem.

### A. Local Planner

In this work, a reaction-based local planner is proposed to give quick reactions to the sensor observations and generate smooth, dynamically-feasible, and collision-free trajectories without relying on explicit map. Considering that (i) the neural networks are skilled in discovering patterns from imprecise or abstract data while the kinodynamic model of UAV is explicit, (ii) the feasible trajectories are in general multi-modal, and (iii) the coupled high-dimensional trajectory representations of polynomial waypoints or B-spline control points is difficult to learn, we propose a primitive-based planner instead of utilizing the expert in [15] or other SOTA classic methods [6], [19]. After that, the LPNet is developed to predict the action-values of primitives instead of specific parameters of trajectory or low-level control commands.

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

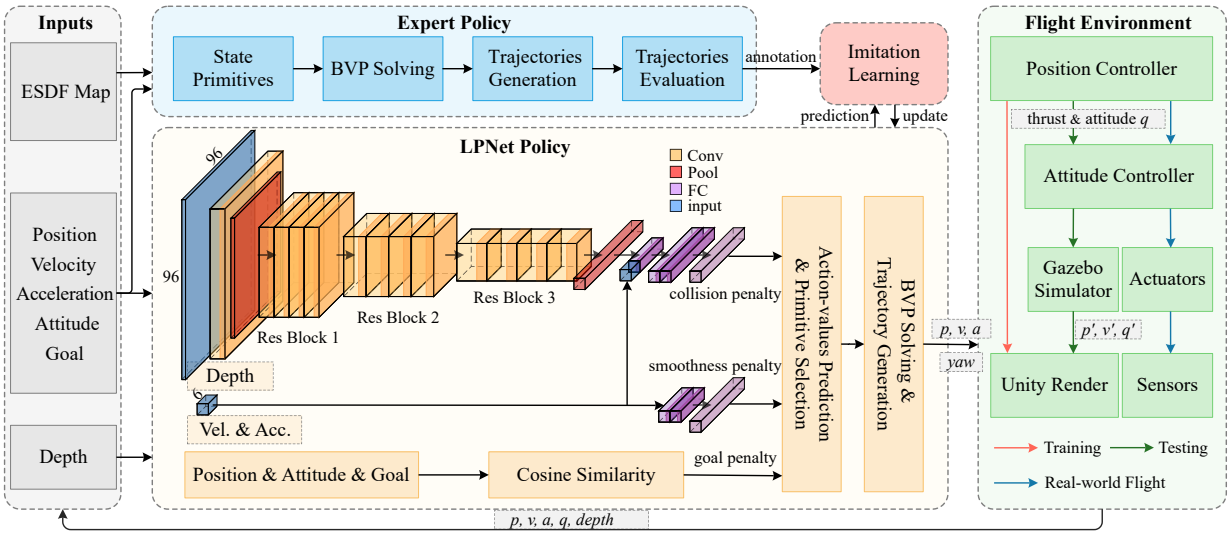



Fig. 2. System overview. In this work, a learning-based local planner, termed as LPNet, is proposed to predict the action-values of motion primitives directly with onboard sensing, and the optimal one is solved as a polynomial trajectory for low-level control. Besides, an expert policy with privileged information of the map is developed to guide the imitation learning, which is only available in the training stage.

*1) Primitive Library:* As visualized in Fig. 3, we construct the primitive library $\mathcal{A} = \{a_0, a_1, ..., a_{N-1}\}$ in state lattice space. Each node represents a 6-dimensional state including position and velocity, and is heading discretized to guarantee that the trajectories always lie in the visible range of depth sensor. Let the horizontal and vertical field of planning denoted by $\theta_x$ and $\theta_y$ respectively, we sample $N_i$ and $N_j$ positions in each direction with uniform heading angles. Assuming that the velocity of quadrotor in flight is close to the desired speed, it is optional to sample $N_k$ velocity directions with fixed final velocity $\|\mathbf{v_k}\| = v_{des}$ to generate trajectories with various curvatures. As illustrated in Fig. 3, the state of primitive $a_n = (\mathbf{p}_{ij}, \mathbf{v}_k) \,|\, n = i \cdot N_j \cdot N_k + j \cdot N_k + k$ is defined as

$$\begin{aligned} \mathbf{p}_{ij} &= (r\cos\varphi_j \cos\psi_i,\ r\cos\varphi_j \sin\psi_i,\ r\sin\varphi_j) \\ \mathbf{v}_k &= (v_{des}\cos(\psi_i + \omega_k),\ v_{des}\sin(\psi_i + \omega_k),\ 0) \end{aligned} \tag{1}$$

where $r$ is the planning horizon, $v_{des}$ is the desired flight velocity, $\psi_i$ and $\varphi_j$ are the sampled angles in horizontal and vertical, which can be presented by

$$\begin{aligned} \psi_i &= \theta_x\Big(\frac{i}{N_i - 1} - \frac{1}{2}\Big),\ i \in [0, N_i) \\ \varphi_j &= \theta_y\Big(\frac{j}{N_j - 1} - \frac{1}{2}\Big),\ j \in [0, N_j)\,. \end{aligned} \tag{2}$$

$\omega_k$ is the sampled angles of velocity in $xy$-plane with the resolution of $\Delta\omega$:

$$\omega_k = \Big(\frac{1 - N_k}{2} + k\Big)\Delta\omega,\ k \in [0, N_k)\,. \tag{3}$$

Considering the differential flatness of the quadrotor dynamics when subject to linear rotor drag effects, we represent the primitive trajectory as three independent 1-D time-parameterized polynomials in the body frame:

$$\mathbf{p}(t) = \sum_{m=0}^{M} \mathbf{k}_m t^m,\ \mathbf{p}(t) = [p_x(t), p_y(t), p_z(t)]^T\,. \tag{4}$$

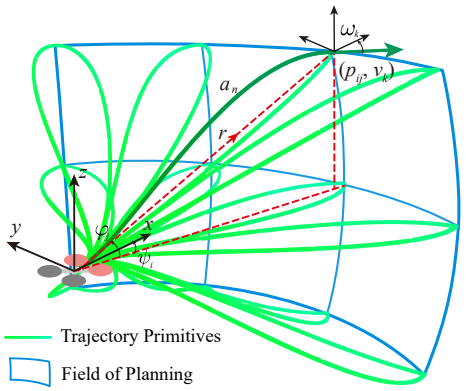



Fig. 3. Illustration of 4×3×2 primitive library: We sample primitives in state lattice (the blue grid) in the view field of sensor and generate the minimum jerk trajectories (the green curves) by solving the boundary value problem.

The goal of the primitive generator is to compute the coefficients of polynomials which guide the quadrotor from the current posture $(\mathbf{p}_0, \mathbf{v}_0, \mathbf{a}_0)$ to the desired end states $(\mathbf{p}_{ij}, \mathbf{v}_k)$ in final time $T$, while minimizing the cost function:

$$C = \frac{1}{T}\int_0^T \|\mathbf{u}(t)\|^2 dt\,. \tag{5}$$

In the presented work, the quadrotor is modeled as a linear dynamic system described by (6) with jerk as control input.

$$\begin{aligned} \dot{\mathbf{X}} &= \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{u} \\ \mathbf{A} &= \begin{bmatrix} \mathbf{0} & \mathbf{I_3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I_3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I_3} \end{bmatrix} \end{aligned} \tag{6}$$

where $\mathbf{X} = [\mathbf{p}(t)^T, \dot{\mathbf{p}}(t)^T, \ddot{\mathbf{p}}(t)^T]^T$ is the state vector and the control input $\mathbf{u} = \dddot{\mathbf{p}}(t)$. Finally, the closed form trajectory can be obtained by applying the Pontryagins minimum principle and substitute the boundary constraints as [20]:

$$\mathbf{p}(t) = \frac{\alpha}{120}t^5 + \frac{\beta}{24}t^4 + \frac{\gamma}{6}t^3 + \frac{\mathbf{a}_0}{2}t^2 + \mathbf{v}_0 t + \mathbf{p}_0 \tag{7}$$

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

where

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{T^5} \begin{bmatrix} 320 & -120T \\ -200T & 72T^2 \\ 40T^2 & -12T^3 \end{bmatrix} \begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{v} \end{bmatrix} \tag{8}$$

and

$$\begin{bmatrix} \Delta\mathbf{p} \\ \Delta\mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{ij} - \mathbf{p}_0 - \mathbf{v}_0 T - \frac{1}{2}\mathbf{a}_0 T^2 \\ \mathbf{v}_k - \mathbf{v}_0 - \mathbf{a}_0 T \end{bmatrix}. \tag{9}$$

For faster velocity response, the execution time $T$ is simply defined as

$$T = \frac{2r}{\|\mathbf{v}_0\| + \|\mathbf{v}_k\|} \tag{10}$$

instead of the optimal time that minimizes the cost in (5).

*2) Network Architecture:* Given the predefined state primitives, the trajectory generation problem is formulated as the action-value evaluation of each primitive:

$$Q^*(s,a) = -\hat{J} = -(\lambda_c \hat{J}_c + \lambda_s \hat{J}_s + \lambda_g \hat{J}_g) \tag{11}$$

where $s$ is the airborne sensor observations, $\hat{J}_c$ is the collision penalty to penalize the distances to obstacles, $\hat{J}_s$ is smoothness penalty to minimize high order derivatives and make the trajectory smooth, $\hat{J}_g$ is the goal penalty to drive the quadrotor towards the goal, and $\lambda_c, \lambda_s, \lambda_g$ are the adjustable weights. The specific definitions of penalties are given in III-B1.

As visualized in Fig. 2 and Table. I, a lightweight network is designed to approximate the desired action-value function $Q^*(s,a)$ with observations of depth image, current velocity, and acceleration. The position and attitude of quadrotor are unnecessary since the primitives and observations are all in the body frame. To minimize the gap of the perception from simulation to reality for more efficient policy transfer, Semi-Global Matching (SGM) is adopted to compute the depth from simulated stereo cameras, as shown in Fig. 4(c). For computational efficiency, we apply the nearest neighbor interpolation algorithm to filter the invalid values and down-sample the depth images to $96 \times 96$ resolution. Since the scales of observations are inconsistent, we standardize the inputs to improve the convergence properties of network by

$$s_i = \frac{s_i - \mu_i}{\sigma_i} \tag{12}$$

where $s_i \subset \{\mathbf{v}_0, \mathbf{a}_0\}$, $\mu_i$ and $\sigma_i$ are the mean and standard deviation. Besides, the depth values are truncated at the detection range of $[0, 10]$m and normalized to $[0, 1]$ by Min-Max scaling.

The architecture of LPNet consists of two branches that predict the collision penalty and smoothness penalty respectively. It is workable to produce the final action-values of (11) directly in a single branch. However, the incorporating policy complicates the potential patterns between inputs and outputs, while the separate strategy is more specific. On the other hand, predicting the penalties separately facilitates the adjustment of weights $\lambda$. As visualized in Table. I, we compress ResNet-18 [21] further by removing the last two residual blocks and performing down-sampling in the first block. We utilize the modified network, refer to as ResNet-14, to extract low-dimensional feature embeddings of the depth image. After that, the depth features together with the velocity and acceleration are processed by a three-layer perception with $[256, 256]$ hidden layers to generate the collision penalty $J_c \subset \mathbb{R}^N$. Another perceptron with $[64, 64]$ hidden layers is adopted to predict the smoothness penalty $J_s \subset \mathbb{R}^N$ with inputs of velocity and acceleration. Moreover, the goal penalty $J_g \subset \mathbb{R}^N$ is defined as the cosine similarity between the heading angle of primitives and desired direction, which can be calculated explicitly by (17). Compared with the end-to-end regression of optimal trajectories which treats the loss function as the disparity from optimal ones, the prediction of costs considers the multi-modal nature of the navigation task. That is, there are many valid trajectories avoiding the obstacles and the costs of which are equally acceptable. Finally, the primitive with optimal action-value (or minimum cost, equivalently) is represented as a polynomial trajectory (7) by solving the optimal control problem and transformed into world frame for low-level control in a receding horizon fashion. It is worth noting that all primitives are parallel evaluated in single forward propagation and only the optimal trajectory is solved. That is, the computation of the proposed approach is slightly affected by the number of primitives while increases linearly in classical methods.

TABLE I
NETWORK ARCHITECTURE

| Operator | Input | Filters | Output |
|---|---|---|---|
| convolutional | $96 \times 96 \times 1$ | $7 \times 7, 64$, stride 2 | $48 \times 48 \times 64$ |
| max pooling | $48 \times 48 \times 64$ | $3 \times 3$, stride 2 | $24 \times 24 \times 64$ |
| residual block1 | $24 \times 24 \times 64$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$ | $12 \times 12 \times 64$ |
| residual block2 | $12 \times 12 \times 64$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$ | $6 \times 6 \times 128$ |
| residual block3 | $6 \times 6 \times 128$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$ | $3 \times 3 \times 256$ |
| average pooling | $3 \times 3 \times 256$ | $3 \times 3$ | $1 \times 1 \times 256$ |
| fully connected | 256 | 64 | 64 |
| collision branch | $64 + 6$ | $[256, 256, N]$ | $N$ |
| smooth branch | 6 | $[64, 64, N]$ | $N$ |

### B. Training Method

Compared with the direct association between inputs and targets in supervised learning, the reward signal of reinforcement learning algorithms (such as deep Q-learning for the proposed task) is frequently noisy and delayed, and the feedback about an action may only be reflected after dozens of time-steps elapsed. Therefore, for faster convergence, a classical lattice planner $Q^*(s^*, a)$ with unconstrained computational budget and privileged observations $s^*$ is treated as the expert policy to guide the imitation learning.

*1) Expert Policy:* In this work, the expert policy is defined as a primitive-based planner with privileged observations of the ground truth of ESDF map and state information. The navigation problem is formulated as primitives evaluation that takes safety, smoothness, and goal into account.

**Collision penalty:** The collision penalty is utilized to penalize the trajectory close to the obstacles. The cost function

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

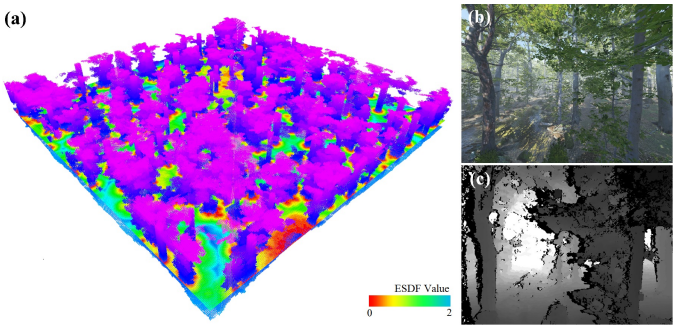



Fig. 4. Observations of environment. (a) is the privileged information of ESDF map provided for the expert policy in training, while only the sensor measurements of RGB (b) and depth (c) image are available for LPNet.

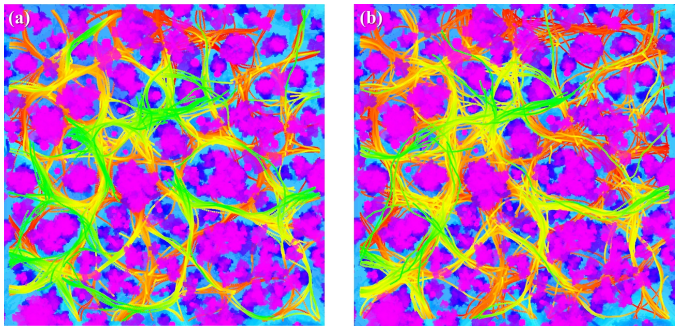



Fig. 5. Executed trajectories in fine-tuning. Compared with the greedy strategy (left), the trajectories executed by $\epsilon$-greedy strategy (right) are more dispersed, which is beneficial to prevent overfitting in training process.

is formulated as the future discounted penalty as (13) since the trajectory is executed in a receding horizon fashion:

$$\hat{J}_c = \frac{\varepsilon^T - 1}{\ln \varepsilon} \int_0^T \varepsilon^t F(d(t)) dt \tag{13}$$

where $\varepsilon \in (0,1)$ is the discounted factor, $(\varepsilon^T - 1)/\ln \varepsilon = 1/\int_0^T \varepsilon^t dt$ is used for averaging, $d(t)$ is distance between the quadrotor and the closest obstacle at time step $t$, and $F(\cdot)$ is expressed by

$$F(d) = \begin{cases} 0 & d \geq d_{thr} \\ (d - d_{thr})^2 & d < d_{thr} \end{cases} . \tag{14}$$

The collision costs of positions are ignored when it is away from obstacles more than $d_{thr}$. To query the distance $d(t)$, a Euclidean Signed Distance Field (ESDF) map is constructed in advance based on the ground truth of the point cloud map, as visualized in Fig. 4(a).

**Smoothness penalty:** The smoothness penalty is to constrain the high order derivatives and ensure the feasibility of the trajectory. In this work, the smoothness penalty $\hat{J}_s$ is formulized as the integral over square jerk of the trajectory same as (5):

$$\hat{J}_s = \frac{1}{T} \int_0^T \dddot{p}^2(t) dt \,. \tag{15}$$

Furthermore, the cost value can be explicitly calculated by

$$\hat{J}_s = \gamma^2 + \beta\gamma T + \frac{1}{T}\beta^2 T^2 + \frac{1}{3}\alpha\gamma T^2 + \frac{1}{4}\alpha\beta T^3 + \frac{1}{20}\alpha^2 T^4. \tag{16}$$

**Goal penalty:** To guide the quadrotor flying towards the goal, the goal penalty $\hat{J}_g$ is designed as the cosine similarity between the heading angle of primitives and reference direction:

$$\hat{J}_g = \frac{\mathbf{d}_p \cdot \mathbf{d}_g}{\|\mathbf{d}_p\| \, \|\mathbf{d}_g\|} \tag{17}$$

where $\mathbf{d}_p$, $\mathbf{d}_g$ represent the direction vectors of primitives and goal, respectively. In the learning-based policy, the goal penalty is also calculated by (17) directly because the directions of primitives are constant in body frame.

Finally, the penalties are standardized to independent distributions with mean of 0 and variance of 1, which benefits the adjustment of parameters in (11) and the convergence in early training stage.

*2) Training Strategy:* In this section, the goal is to train the network policy $Q(s,a)$ approach to the privileged expert $Q^*(s^*,a)$. Therefore, the loss function is defined as the average Smooth L1 loss of the penalties of all primitives:

$$\mathcal{L} = \frac{1}{N} \sum_{n=0}^{N-1} L_1(\hat{J}_c^n - J_c^n) + L_1(\hat{J}_s^n - J_s^n) \tag{18}$$

where the superscripts $n$ represents the $n$-th primitives and $L_1(\cdot)$ is defined as a piecewise function of error $\Delta J$:

$$L_1(\Delta J) = \begin{cases} 0.5\Delta J^2 & |\Delta J| < 1 \\ |\Delta J| - 0.5 & otherwise \end{cases} . \tag{19}$$

The proposed LPNet is trained in Flightmare [22] simulator. For sufficient coverage of the observation space and preventing crashes in the early training stage, a dataset of observations and expert annotations is collected to train the LPNet from scratch. We collect 100k samples by randomly resetting the pose of quadrotor and iterate 50 epochs for pre-training with the learning rate of 0.0003 and batch size of 128. After that, we fine-tune the pre-trained model for another 50k time steps using the strategy combining experience replay and dataset aggregation to assure the distribution of observations consistent with real-flight. During the training loop, we aggregate samples into replay memory by rolling out the current policy and annotating the action-values of primitives with the expert policy simultaneously. To ensure adequate exploration of scenario, we execute the action according to $\epsilon$-greedy strategy that follows the greedy strategy with probability $1-\epsilon$ and selects a random action otherwise, as shown in Fig. 5. Note that the goal penalty is set to zero in training stage to let the quadrotor fly freely. Every 20 steps we apply 2 minibatch updates to samples of experience drawn at random from the data pool. As shown by the red arrow in Fig. 2, it is unnecessary to simulate the physical interactions such as control noise, inference latency, and model uncertainty because the outputs of LPNet are kinodynamic-free, making it more robust for Sim-to-Real transfer.

## IV. Experiment

In this section, a series of experiments are conducted to verify the proposed learning-based local planner. We run a number of ablations to analyze the proposed LPNet and compare it against the state-of-the-art methods in simulation

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

on i7-9700 CPU and RTX 3060 GPU. Furthermore, the proposed method is implemented on a physical quadrotor with the computational unit of NVIDIA Xavier NX for real-world experiment. More details are available in the attached video at https://youtu.be/4PaDz5MERtE.

### A. Ablation Experiment

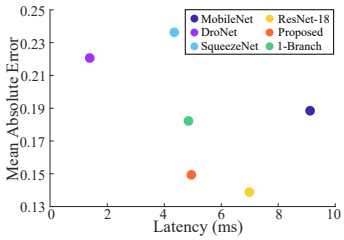
*1) Network Architecture:* We investigate the mean absolute error and time consumption of the proposed ResNet-14 with other lightweight backbones DroNet, MobileNet-v3, SqueezeNet, and ResNet-18. As visualized in Fig. 6(a), the modified ResNet-14 exhibits better accuracy than most baselines on this task while keeping lightweight. Besides, it is workable to predict the final action-values directly in a single branch. Therefore, we compare the performance of the incorporating policy (the green scatter) with the separate strategy (the red scatter) in Fig. 6(a), where the proposed separate strategy achieves about 33% accuracy improvement without sacrificing real-time performance. The result indicates that predicting the collision and smoothness penalty respectively makes the pattern between inputs and outputs more specific, and therefore, facilitates the convergence of network.

*2) Input Source:* To minimize the domain shift from simulation to reality, a stereo matching algorithm SGM is adopted to compute the depth images from simulated stereo cameras and down-sampled to $96 \times 96$ for time efficiency. In Fig. 6(b), we compare the network performance with different input sources, including SGM images, ground truth images, high resolution images ($224 \times 224$), RGB-D images, and RGB images. As visualized, the network with SGM inputs exhibits similar accuracy with high resolution images and the ground truth images, which indicates that this representation is rich enough for trajectories evaluation in complex environment. On the other hand, it also reflects that the data-driven approach can leverage the regularity of inputs and be more robust to the sensor noise. In addition, considering the gap of RGB images between simulation and reality, we do not utilize RGB-D images although they achieve the best results.

*3) Primitive Number:* The primitive library can be modified according to the scenario and only the output layers need to be revised. All action-values are predicted in one propagation parallelly and only the trajectory with optimal action-value will be solved. Therefore, contrary to classical primitive-based methods with linear time complexity, the runtime performance of the proposed network is insensitive to the number of primitives. Nevertheless, the accuracy performance will decrease because the capacity of neural network is limited. The ablation results are shown in Fig. 6(c), where the safety metrics are evaluated by the average distance to obstacles in simulated experiments, and the computational cost is measured by MFLOPs (million floating point of operations).

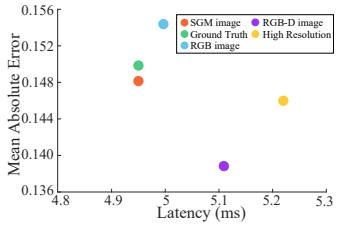
### B. Comparison Experiment

In this section, we compare the proposed approach with three state-of-the-art methods: Fast Planner [6], EGO Planner [23], and our previous work MPPI Planner [8] as baselines. In proposed method, we sample $19 \times 15 \times 3 = 405$ primitives



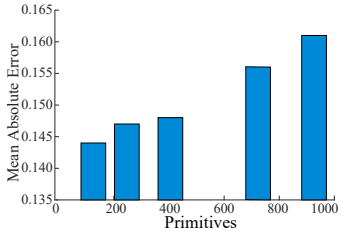

| Architecture | MAE | Latency[$\mu/\sigma$] | Params |
|---|---|---|---|
| MobileNet | 0.184 | 9.12/0.024 | 2.54M |
| DroNet | 0.221 | 1.35/0.032 | 1.41M |
| SqueezeNet | 0.235 | 4.35/0.031 | 1.25M |
| ResNet-18 | 0.135 | 6.97/0.043 | 11.7M |
| Proposed | 0.148 | 4.95/0.018 | 2.96M |
| 1-Branch | 0.181 | 4.80/0.076 | 2.94M |

(a) Network Architecture




| Input Source | MAE | Latency[$\mu/\sigma$] |
|---|---|---|
| SGM Image | 0.148 | 4.95/0.018 |
| Ground Truth | 0.150 | 4.95/0.020 |
| RGB Image | 0.154 | 5.00/0.025 |
| RGBD Image | 0.139 | 5.11/0.011 |
| HiRes Image | 0.146 | 5.22/0.013 |

(b) Input Source




| Primitives | MAE | Latency[$\mu/\sigma$] | Safety(m) |
|---|---|---|---|
| 135(15×9×1) | 0.144 | 4.79/0.020 | 1.12 |
| 247(19×13×1) | 0.147 | 4.86/0.018 | 1.29 |
| 405(15×9×3) | 0.148 | 4.95/0.021 | 1.33 |
| 741(19×13×3) | 0.156 | 4.96/0.027 | 1.17 |
| 945(21×15×3) | 0.161 | 4.99/0.022 | 1.06 |

(c) Primitive Number

Fig. 6. Ablation experiment. Where the mean $\mu$ and standard deviation $\sigma$ of latency is computed over 1000 samples.

with the resolution of $3.95° \times 3.67° \times 45°$ in horizon, vertical, and speed direction respectively. The average performance statistics and the qualitative results are illustrated in Table II and Fig. 7.

*1) Processing Latency:* In this section, we compare the computational cost of our approach against the baselines. In Table II, the mapping process consists of depth raycasting, grid map construction and ESDF generation (if exists), and the planning process is trajectory evaluation and optimization. To clarify, the baselines are evaluated with depth resolution at $320 \times 240$, raycasting length of 7.5 m, updating size of $7.5 \times 7.5 \times 5.5$ m$^3$, and voxel size of 0.1 m for better safety and real-time performance trade-offs in testing environment. With total computation time of 64.6 ms per frame, Fast Planner results in the highest processing latency, most of which is spent on ESDF computation. The results can be attributed to the cluttered projected point cloud in unstructured scenario, which leads larger ESDF updating size ($7.2 \times 11.5 \times 7.8$ m$^3$ in average) for safety navigation. Besides, raycasting and occupancy probability integration are leveraged in mapping to reject outliers of the depth sensor, making it more robust to the noise but more time-consuming. Implemented parallel on GPU, MPPI Planner significantly reduces the mapping time and achieves the minimum latency in the baselines. However, it predicts the optimal trajectory with a Monte Carlo approximation using forward sampling, which is computationally complex and takes 7.7 ms in trajectory generation. By contrast, our approach achieves the best real-time performance in total, which only takes 5.0 ms for network inference on average.

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

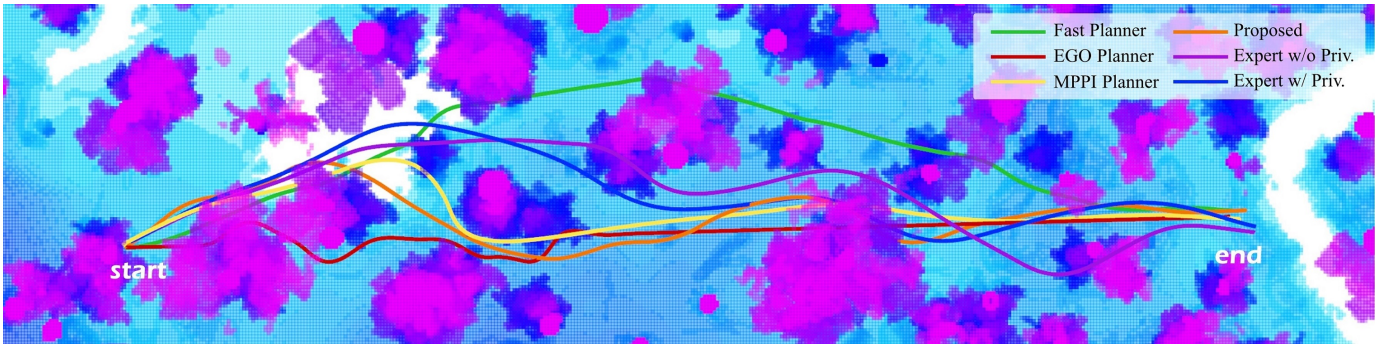



Fig. 7. Trajectory comparisons in simulated forest experiment with massive trees and low bushes.

TABLE II
QUANTITATIVE COMPARISON

| Method | Processing Latency | | | Trajectory Comparison | | | |
|---|---|---|---|---|---|---|---|
| | Mapping (ms) | Planning (ms) | Total (ms) | Flight Time (s) | Traj. Length (m) | Safety Distance (m) | Energy ($m^2/s^5$) |
| Fast Planner | 62.5 | 2.1 | 64.6 | 30.5 | 52.7 | 0.98 | 21.6 |
| EGO Planner | 14.4 | 0.5 | 14.9 | 30.7 | **50.9** | 0.94 | 56.7 |
| MPPI Planner | 6.5 | 7.7 | 14.2 | 28.9 | 53.6 | 1.09 | **13.2** |
| Proposed | / | 5.0 | **5.0** | **28.1** | 52.5 | **1.33** | 25.5 |
| Expert w/o Priv. | 16.7 | 8.2 | 24.9 | 29.2 | 53.6 | 1.07 | 32.7 |
| Expert w/ Priv. | / | / | / | 28.7 | 53.7 | 1.49 | 22.2 |

* The results in bold are the optimal values among different methods except the privileged expert.

On the one hand, we reformulate the planning problem as an end-to-end regression and save the computation of mapping in classical methods. On the other hand, compared with the sampling-based method [8], we spare the effort of massive sampling and only the trajectory with optimal action-value will be solved.

*2) Trajectory Comparison:* In this section, we conduct a set of experiments in a simulated forest with massive trees and bushes to evaluate the navigation performance of our approach. In this experiment, the goal is set to 50 m away from the quadrotor and the desired flight speed is 2 m/s. For a fair comparison, we provide the depth ground truth as the input of baselines for accurate mapping. As illustrated in Table II, MPPI and EGO Planner achieve the minimum energy cost (integral of the squared jerk) and the shortest trajectory length respectively, while the proposed method outperforms the baselines in the average distance to obstacles. The drop in safety distance of the baselines can be attributed to the latency of mapping, which makes it challenging for high-speed flight in obstacle-dense environment. In addition, occupancy probability integration is essential to remove the sensor noise, which introduces extra latency of mapping because multiple observations are required to completely add the obstacles to the map. In contrast, the proposed method exhibits better real-time performance by evaluating the action-values directly from sensor measurements without mapping. The neural network is able to extract abstract features and leverage the regularities of training data, making it more robust to the sensor noise. Furthermore, the primitive trajectories have larger search scope and are not restricted to the local optimal solution around the guiding path. However, without detailed geometric map, the network tends to choose the safe and conservative trajectory from finite primitive library, and therefore, the proposed approach incurs higher control cost and trajectory length.

Additionally, we evaluate the performance of the expert policy with/without privileged information. As visualized in Table II, the privileged expert is competitive compared with the SOTA methods and demonstrates the best safety performance thanks to the ground truth of map. Benefiting from the data-driven nature of neural networks and the noise robustness of deep features, the network policy outperforms the non-privileged expert in most metrics. Therefore, we can conclude that the proposed network policy is not only a lightweight version of the expert, but also performs similarly to the privileged expert while only relying on airborne sensor observations.

### C. Real-world Experiment

In this section, the proposed method is implemented on a small quadrotor platform with 250 mm diameter and 1.15 kg mass, as illustrated in Fig. 8(a). The main computational unit is an NVIDIA Xavier NX with 6-core ARM CPU and 384 CUDA cores GPU, on which the proposed approach can run at 18.2 FPS on average. Sensing is performed by Intel RealSense T265 and RealSense D435i. The RealSense T265 is a commercial off-the-shelf tracking camera that runs a visual-inertial odometry to output the state estimation of the platform. The RealSense D435i is adopted to provide the depth images for our reaction-based local planner with $87^\circ \times 58^\circ$ FOV and around 10 m sensor range. The LPNet is utilized to predict the action-values and generate the dynamically feasible trajectory from sensor measurements. After that, the reference trajectory is tracked by a geometric controller and the open-source autopilot Pixhawk on the quadrotor.

As depicted in Fig. 8(b), the real-world flight experiment is conducted in a previously unknown forest, which is challenging for autonomous flight online. In this experiment, we sample $35 \times 7 \times 1 = 245$ primitives with the resolution of $2.14^\circ \times 2.85^\circ$. As demonstrated in Fig. 9, the drone executes

**IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.**

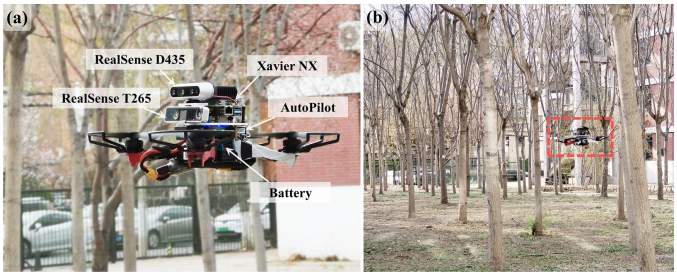



Fig. 8. Real-world experiment. (a) Illustration of our physical platform. (b) The forest environment of the real-world flight.

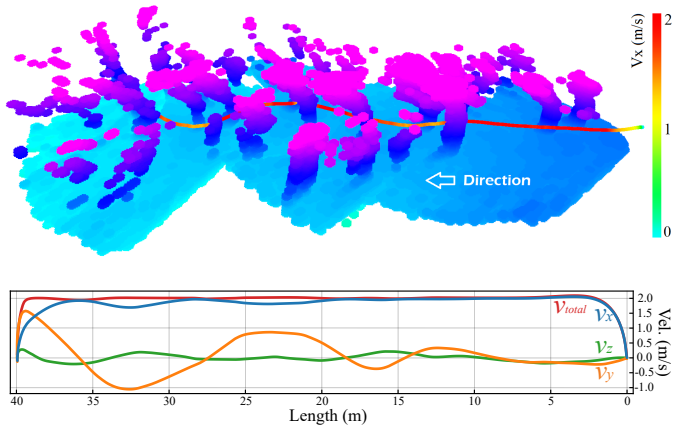



Fig. 9. Trajectory visualization of the real-world flight experiment in dense forest with velocity profile. Note that the map is only constructed for demonstration.

an aggressive trajectory through the dense forest at the average speed of 2.0 m/s. Note that the map is only constructed for visualization and is unavailable for the quadrotor. The experiment validates that the proposed end-to-end local planner is capable of performing autonomous navigation flight while avoiding obstacles.

## V. Conclusions

In this work, we proposed a learning-based local planner for autonomous collision avoidance of quadrotors without explicitly mapping. It is applicable for trivial obstacles or with reference waypoints given by the high-level policy in structured scenarios. Firstly, a lightweight network is developed to predict the action-values of motion primitives from sensory observations directly. Then, the optimal primitive is represented as a polynomial trajectory by solving the optimal control problem. The network is trained by imitating an expert policy with privileged information of map and unconstrained computational budget. Finally, a set of comparisons and autonomous flight experiments are conducted to validate the proposed policy in simulation and real-world.

## References

[1] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*, pp. 2520–2525. IEEE, 2011.

[2] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: The 16th International Symposium ISRR*, pp. 649–666. Springer, 2016.

[3] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation on point clouds," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 139–146. IEEE, 2016.

[4] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[5] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 3681–3688. IEEE, 2017.

[6] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

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

[8] H. Lu, Q. Zong, S. Lai, B. Tian, and L. Xie, "Flight with limited field of view: A parallel and gradient-free strategy for micro aerial vehicle," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 9, pp. 9258–9267, 2021.

[9] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 2872–2879. IEEE, 2017.

[10] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 179–185. IEEE, 2020.

[11] L. Han, F. Gao, B. Zhou, and 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)*, pp. 4423–4430. IEEE, 2019.

[12] F. Sadeghi and S. Levine, "CAD2RL: Real single-image flight without a single real image," in *Robotics: Science and Systems(RSS)*, 2017.

[13] C. Xiao, P. Lu, and Q. He, "Flying through a narrow gap using end-to-end deep reinforcement learning augmented with curriculum learning and sim2real," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[14] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[15] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[16] J. Tordesillas and J. P. How, "Deep-panther: Learning-based perception-aware trajectory planner in dynamic environments," *IEEE Robotics and Automation Letters*, 2023.

[17] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[18] H. Nguyen, S. H. Fyhn, P. De Petris, and K. Alexis, "Motion primitives-based navigation planning using deep collision prediction," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9660–9667. IEEE, 2022.

[19] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[20] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[22] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, pp. 1147–1157. PMLR, 2021.

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