End-to-End Depth-Based Autonomous Obstacle Avoidance for UAVs Using Hybrid Imitation and Reinforcement Learning

1st Hairui Yin
Science Academy
University of Maryland, College Park
College Park, United States
yinhr@umd.edu

2nd Javad Baghirov
Science Academy
University of Maryland, College Park
College Park, United States
jbaghiro@umd.edu

3rd Hsing-Hao Wang
Science Academy
University of Maryland, College Park
College Park, United States
hwang138@umd.edu

***Abstract*—Our project present a lightweight, end-to-end autonomous obstacle avoidance system for unmanned aerial vehicles (UAVs) operating in mapless environments such as indoor corridors and densely cluttered spaces. The proposed framework relies on a front-facing depth camera, a down-facing depth camera, and a policy with reinforcement learning (RL) fine-tuning that directly maps raw perception inputs to body-frame velocity commands, achieving a closed-loop perception–decision–control architecture. To address the difficulties of reinforcement learning (RL) in complex environments, including unsafe exploration, low sample efficiency, and unstable early training, we adopt a hybrid Imitation Learning (IL) + Reinforcement Learning (RL) strategy. First, a rule-based expert system is designed to provide high-quality demonstrations in the simulation, enabling the policy network to acquire a baseline navigation and collision-avoidance skills through supervised learning. Then, the pretrained model is fine-tuned using RL algorithms (PPO) with a reward structure that encourages fast and safe traversal, allowing the agent to surpass the limitations of the expert policy. We integrated into the Gazebo and PX4 simulation environment to produces large-scale datasets for efficient policy training. Experimental results demonstrate that the hybrid learning approach yields smooth, robust flight trajectories and significantly improves obstacle-avoidance performance.**

I. Introduction

Autonomous navigation in unknown environments remains a challenge for unmanned aerial vehicles (UAVs). While recent advances in simultaneous localization and mapping (SLAM), depth sensing, and model-based planning have significantly improved UAV autonomy, these approaches typically rely on accurate state estimation or preconstructed maps. Many traditional planning pipelines are built from separate perception, mapping, and control modules that require extensive tuning and careful integration. As a result, these systems often struggle with sensor noise, moving obstacles, or environments they haven't encountered before.

End-to-end learning-based navigation offers a promising alternative by directly mapping raw sensory inputs to control actions without explicit mapping or handcrafted planning rules. However, fully learning-based approach has its own challenges. Reinforcement learning (RL) methods suffer from extremely low sample efficiency, unstable convergence, and high crash rates during early exploration issues. On the other hand, supervised imitation learning (IL) is limited by the quality and diversity of expert demonstrations, often causing learned policies to inherit suboptimal behaviors and struggle with out-of-distribution scenarios.

Fig. 1: Image of the drone in our environment

To address these limitations, we proposes a lightweight, robust, and scalable end-to-end obstacle-avoidance system that enables UAVs to navigate safely in complex and mapless environment with a front-facing and down-facing depth camera. The core idea is a hybrid learning framework that leverages the advantages of both imitation learning and reinforcement learning. We first design a rule-based expert system capable of producing consistent, high-quality control demonstrations in simulation. Through supervised behavior cloning, the neural policy network acquires basic navigation and collision-avoidance capabilities, effectively solving the cold-start problem of RL. The pretrained policy is then fine-tuned using an RL algorithm with a reward function that encourages safe, smooth, and efficient traversal. This two-stage pipeline enables the agent to not only replicate but also exceed the performance of the expert system.

Beyond obstacle avoidance, the system incorporates a

lightweight vision-based landing module to demonstrate precise low-altitude control using minimal sensing. A downward-facing camera and PX4 odometry are used to detect a black landing pad by thresholding dark regions and extracting the largest contour, whose centroid provides pixel-level alignment error. This error is converted into velocity commands via a proportional controller and rotated into the NED frame using the UAV's yaw. A finite state machine governs centering, descent, pause, ascent, and completion, with a blind fallback mode that relies on altitude and timing when visual feedback is lost. This component illustrates how simple rule-based controllers can complement learning-based navigation within a unified autonomous framework.

II. Related Work

A. Vision-based End-to-End Drone Control

Early autonomous navigation frameworks predominantly relied on "map-then-plan" architectures. Optimization-based methods, such as the minimum snap trajectory generation by Mellinger and Kumar [1], provided mathematical guarantees for agility but required precise state estimation. Subsequent works like Ego-Planner [2] integrated gradient-based optimization with grid maps to achieve robust local planning without Euclidean Signed Distance Fields (ESDF), though they remain computationally intensive for resource-constrained MAVs. To reduce latency, research shifted toward reactive navigation, where sensory inputs are directly mapped to control actions. DroNet [3] demonstrated that a residual network could learn to navigate urban environments by imitating the driving behavior of cars, proving the viability of learning from diverse data sources. Building on this, CAD2RL [4] introduced domain randomization to bridge the reality gap, training collision avoidance policies entirely in simulation using single monocular images. More recently, Fan et al. [5] advanced this domain by compressing sparse 3D LiDAR point clouds into 2D obstacle maps, training an end-to-end Deep Reinforcement Learning (DRL) agent to navigate highly dynamic environments with significantly lower latency than rule-based planners.

B. Event Vision and Policy Optimization

High-speed navigation demands sensors capable of handling motion blur and high dynamic range. Event cameras have emerged as a solution, asynchronously capturing brightness changes with microsecond latency. Falanga et al. [6] pioneered dynamic obstacle avoidance using event cameras, leveraging their speed to dodge moving objects. Recently, Bhattacharya et al. [7] extended this to static environments, employing a teacher-student distillation framework where a vision-based student learns from a model-based expert using depth prediction as a pretext task. In parallel, the training methodologies for these policies have evolved. Deep Reinforcement Learning (DRL) has achieved superhuman performance in tasks like drone racing Kaufmann et al. [8], but it typically suffers from sample inefficiency and dangerous exploration phases. Algorithms like Proximal Policy Optimization (PPO) [9] provide stable update steps but still face cold-start instability. Conversely, Imitation Learning (IL), used by Loquercio et al. [10] to train agile flight policies in the wild, offers stable convergence but is limited by the distribution of the expert demonstrations.

Balancing training stability with adaptability remains a critical challenge in learning-based navigation. Pure Reinforcement Learning often suffers from sample inefficiency and hazardous initial exploration, while Imitation Learning is constrained by the fixed distribution of expert data, failing to generalize to novel scenarios. We propose a hybrid framework that synthesizes these paradigms: using Imitation Learning to warm-start a depth-based policy establishes a safe baseline, which is subsequently fine-tuned via Proximal Policy Optimization (PPO). This approach overcomes the limitations of static demonstrations, enabling robust, adaptive obstacle avoidance in complex, unseen environments.

III. Methodology

This section describes the complete design of our lightweight end-to-end UAV obstacle-avoidance framework. The methodology integrates (i) privileged expert-guided imitation learning, (ii) reinforcement-learning–based fine-tuning, and (iii) a computationally efficient perception–action stack designed for onboard deployment.

A. System Overview

Our objective is to learn a control policy:

$$\pi_\theta : D_{1:T} \rightarrow (v_x, \dot{\psi})$$

that maps a short depth-image sequence to forward velocity v_x and yaw-rate $\dot{\psi}$. Height, roll, and pitch stabilization are handled separately by the PX4 flight controller, allowing the learning system to focus exclusively on planar navigation.

The overall system architecture is designed to address the cold-start problem in reinforcement learning by leveraging expert demonstrations, as shown in Figure 2. The pipeline consists of four main components:

1. **Spatiotemporal Encoding:** Compresses continuous depth streams into a history-stacked 20×20 occupancy grid to capture local geometry and motion dynamics.
2. **Expert Demonstration:** Utilizes a rule-based expert with privileged state access to generate high-quality, collision-free reference trajectories in simulation.
3. **Policy Warm-start:** Pre-trains a lightweight CNN policy via Behavior Cloning on expert data to establish a stable baseline behavior.
4. **RL Fine-tuning:** Refines the pre-trained policy using Proximal Policy Optimization (PPO) with a composite reward function, optimizing for safety, speed, and smoothness beyond the expert's capabilities.

This modular structure allows each part of the system to be optimized independently while remaining tightly integrated during execution.

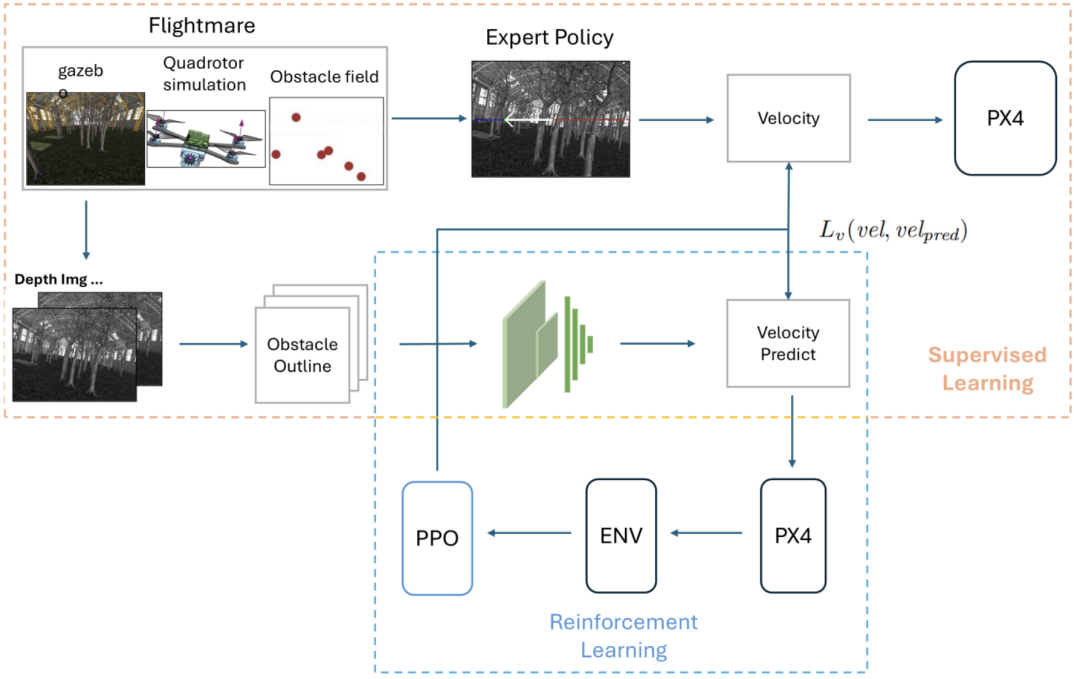

Fig. 2: The overall architecture of our system. The pipeline implements an end-to-end imitation learning framework. Raw visual data is captured from a quadrotor simulated in Gazebo and processed by a CNN-based module to predict depth maps under a depth loss constraint. The resulting obstacle outlines are fused by a neural network to generate the final control velocities. During training, the predicted velocities are supervised via a velocity loss against the expert commands generated by the PX4 flight controller, utilizing an experience buffer for data management.

B. Expert Policy and Data Generation

To mitigate the cold-start and instability issues of reinforcement learning in flight settings [5], we employ a rule-based expert controller. The expert operates with access to privileged information, specifically the precise sector-based obstacle distances derived from the depth sensor, to compute safe waypoints and output continuous velocity commands. This strategy serves as a stable supervisor for imitation learning and a safety reference during reinforcement learning.

1) Simulation Environment: We utilized the Gazebo simulator coupled with the PX4 flight control stack to generate randomized, cluttered obstacle fields. To ensure environmental diversity, the obstacle maps are randomly generated for each episode (see Fig. 3). During expert rollouts, we record synchronized state-action pairs consisting of:

- **Observation:** Temporally stacked local occupancy grids compressed from raw depth images.
- **Action:** Expert control commands (body-frame forward velocity and yaw rate).

2) Expert Control Strategy: The expert policy maps the privileged depth observations to control outputs using a geometric reactive scheme:

- **Safe Corridor Scoring:** Calculating a centering score based on the clearance difference between left and right sectors to steer toward the widest opening.
- **Distance-Based Speed Regulation:** Modulating forward velocity dynamically based on the proximity of the nearest obstacle (e.g., decelerating when $d_{min} < 2.0m$).
- **Virtual Geofence:** Enforcing a soft boundary constraint to keep the drone within the training area by overriding headings when the distance from the origin exceeds a threshold.

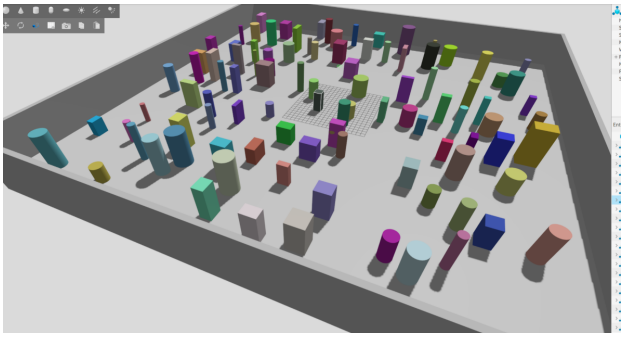

Fig. 3: **Simulation Environment.** We developed a randomized map generation framework capable of supporting the stochastic placement of 200 obstacles within a bounded arena. As shown in the figure, these obstacles vary in height, size, and color, creating a dense and unstructured environment to rigorously test the navigation policy.

C. Lightweight CNN Policy for Reactive Control

We employ a lightweight Convolutional Neural Network (CNN) designed for low-latency inference on edge hardware. The network maps the spatiotemporal occupancy grids directly to control commands, bypassing explicit global path planning.

1) Network Architecture: The actor network consists of a three-layer convolutional encoder followed by a fully connected decision head. The encoder processes the stacked $20\times20\times5$ input grid using 3×3 kernels with batch normalization and ReLU activation, compressing the geometric features into a flat latent vector. The decision head then outputs the continuous action vector $a_t = [v_x, \dot{\psi}]^T$.

2) Imitation Learning Warm-start: To initialize the policy in a safe region, we employ Behavior Cloning (BC) where the student network (CNN) learns to mimic the rule-based expert's actions. Unlike depth supervision approaches [7], our supervision target is the control command itself. The imitation loss L_{BC} is defined as the Mean Squared Error (MSE) between the expert's command a^* and the student's prediction $\hat{a}$:

$$L_{BC} = \frac{1}{N}\sum_{i=1}^{N}\left(||v^*_{x,i} - \hat{v}_{x,i}||^2 + \lambda||\dot{\psi}^*_i - \hat{\dot{\psi}}_i||^2\right)$$

where λ balances the scale difference between linear velocity and yaw rate. This warm-start phase ensures the RL agent begins training with a collision-free policy rather than random exploration.

3) Policy Initialization and Transfer: Instead of freezing a separate perception module, we adopt a *warm-start* strategy for the Reinforcement Learning (RL) phase. Once the Imitation Learning (IL) policy converges, its Convolutional Encoder weights, which have learned to extract spatial features from the occupancy grids, are transferred to initialize the PPO actor network. This design allows the RL agent to inherit a semantic understanding of obstacle geometry immediately, thereby bypassing the sample-inefficient and dangerous random exploration phase typical of training from scratch.

D. Obstacle Map Construction

To ensure real-time performance on embedded hardware while retaining spatial structure, raw depth maps are compressed using a spatiotemporal grid encoding scheme.

1) Spatial Compression (Local Occupancy Grid): The high-resolution depth image is spatially downsampled into a 20×20 local occupancy grid. For each grid cell (r, c), we compute a robust distance value to filter sensor noise. Instead of a simple minimum, we calculate the 5th percentile of the valid depth pixels within the cell:

$$g_{r,c} = \text{Percentile}(\{d_{i,j} \in \text{cell}_{r,c}\}, 5\%)$$

Additionally, a "blind-spot" safety mechanism forces the cell value to 0 if the ratio of invalid (too close) pixels exceeds 40%. The grid values are then clipped to $10m$ and normalized to the range $[0, 1]$.

2) Temporal Stacking: To capture higher-order kinematics (velocity and acceleration relative to obstacles), we concatenate the current grid with the $H = 5$ most recent historical grids. This results in a $5 \times 20 \times 20$ state tensor, which serves as the input to the policy network.

This representation is analogous to a "2.5D" sliding window, allowing the Convolutional Neural Network (CNN) to infer obstacle motion directly from the channel dimension.

E. Control Policy Learning: IL Initialization and RL Fine-tuning

The control network is a lightweight Convolutional Neural Network (CNN) that processes the stacked obstacle grids to output continuous velocity and yaw-rate commands.

1) Imitation Learning (Warm-Start): In the first stage, the CNN is pre-trained via Behavior Cloning (BC) to mimic the expert's rule-based policy. We minimize the Mean Squared Error (MSE) between the student's prediction and the expert's ground-truth commands:

$$L_{BC} = ||\mathbf{a}_{student} - \mathbf{a}_{expert}||^2$$

where $\mathbf{a} = [v_x, \dot{\psi}]^T$. This supervised phase provides a competent, collision-free baseline policy, effectively resolving the sample inefficiency and crash-prone exploration typically associated with "scratch" reinforcement learning.

2) Reinforcement Learning Fine-tuning: The IL-initialized policy is then transferred to a Proximal Policy Optimization (PPO) agent for fine-tuning. This stage allows the policy to optimize for smoothness and speed, surpassing the conservative nature of the expert. The composite reward function R_t is defined as:

$$R_t = R_{\text{Progress}} + R_{\text{Speed}} - (P_{\text{Crash}} + P_{\text{Danger}} + P_{\text{Stability}} + P_{\text{Lazy}})$$

Specific components aligned with our implementation include:

- **Progress Reward:** $20.0 \times \Delta d$, encouraging movement toward the goal (or forward flight).
- **Speed Reward:** $1.5 \times v_x$, incentivizing aggressive but safe velocities.
- **Crash Penalty:** A large penalty followed by an episode reset upon collision ($d_{min} < 0.6m$).
- **Danger Penalty:** A soft penalty when entering the critical buffer zone ($0.6m < d_{min} < 1.2m$).
- **Stability Penalty:** Penalizing high jerk (action differences) to ensure smooth camera footage and protect motors: $0.5 \times (||\Delta v_x|| + ||\Delta\dot{\psi}||)$.

Unlike interactive expert-gating approaches [7], our method relies on PPO's inherent clipping mechanism to prevent the policy from deviating destructively from the safe baseline established during the warm-start phase.

F. Autonomous Landing Module

1) Landing Zone Detection: The bottom-facing camera is thresholded for dark regions. The largest contour is extracted and its centroid computed. Pixel-space error is converted into body-frame velocity via proportional gains, then rotated into NED using PX4 odometry.

2) Finite State Machine (FSM): The landing sequence operates through the states:

- CENTERING – align over pad
- DESCENDING – reduce altitude to target
- PAUSE – stabilize
- ASCEND – return to cruising altitude
- COMPLETE

A fallback BLIND mode activates if camera frames are lost, relying solely on altitude and timers. This module demonstrates the extensibility of the perception control stack to auxiliary UAV behaviors.

G. Scene Reconstruction Module

The system includes an experimental 3D reconstruction module using VGGT [11]. The module:

- converts video sequences into point clouds,
- is capable of coarse environment reconstruction,
- currently suffers from latency (around 5s for 5 frames) and poor scalability (exponential growth with sequence length).

Although not used for onboard obstacle avoidance, this module supports future mapping and memory modules articulated in the project roadmap. The result is shown in Fig. 4.

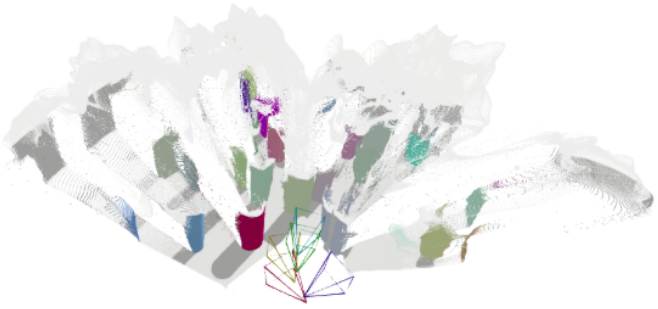

Fig. 4: This figure presents a 3D scene reconstruction produced by VGGT, where sparse point clouds and estimated camera poses recover the global structure of the cluttered environment. The reconstruction captures the spatial layout and relative geometry of obstacles, enabling downstream tasks such as mapping, localization, and motion planning.

H. Real-Time Deployment Architecture

At deployment (inference time), the heavy expert policy and training scaffolds are removed. The agent operates in a streamlined end-to-end loop:

1. **Sensing:** Raw Depth Image $\rightarrow$ Downsampling $\rightarrow 20\times20$ Local Occupancy Grid.
2. **State Construction:** The current grid is pushed into a First-In-First-Out (FIFO) buffer to form the $5 \times 20 \times 20$ spatiotemporal tensor.
3. **Inference:** Stacked Tensor $\rightarrow$ Trained CNN Policy $\rightarrow$ Body-frame velocity commands $(v_x, \dot{\psi})$.
4. **Control:** The Python node computes altitude corrections (P-control) and converts body-frame velocities to global NED frame.
5. **Actuation:** PX4 executes the velocity setpoints via its internal cascade PID controller, maintaining stability while tracking the RL agent's guidance.

The system maintains extremely low latency because the direct mapping from occupancy grids to control eliminates the need for expensive SLAM, global path planning, or intermediate depth-estimation networks.

IV. Contributions

No changes were made to these packages, only worked on top of it.

A. Repositories Used

- **PX4-Autopilot**
  Repository: https://github.com/PX4/PX4-Autopilot
  Branch: `origin/release/1.15`
  Usage: Flight control, offboard control interface.
- **px4_msgs**
  Repository: https://github.com/PX4/px4_msgs
  Branch: `origin/main`
  Usage: ROS 2 message definitions for PX4 communication.
- **ros_gz (ros_gz_bridge)**
  Repository: https://github.com/gazebosim/ros_gz
  Branch: `origin/humble`
  Usage: ROS 2–Gazebo Harmonic bridge for sensor data exchange.
- **Micro-XRCE-DDS-Agent**
  Repository: Micro-XRCE-DDS-Agent GitHub
  Branch: `origin/master`
  Usage: Communication between PX4 and ROS 2.
- **VGGT**
  Repository: https://github.com/facebookresearch/vggt
  Usage: Used as a reference for a similar system design and methodology.

V. Results

A. Training Phase and Imitation Learning

We first evaluate the training process of the base model, which is supervised by an imitation learning loss function designed to clone the expert's behavior. As illustrated in Fig. 5, the model was trained for 360 epochs. To ensure stable convergence and escape local minima, we employed a step-decay learning rate scheduler.

The loss curves demonstrate effective knowledge transfer: both training and validation losses decrease significantly in the initial phase and steadily converge to a minimal value. This low final loss indicates that the base model has successfully learned the latent control policy of the expert, providing a solid initialization for subsequent reinforcement learning.

Metric	Accident Rate =	Speed ↑	Stability ↑	Distance ↑
Expert Policy	0.010	1.89	0.2729	113.5
Base Model	**0.006**	1.98	0.2972	119.53
RL-Fine-Tuned Model	0.2	**2.314**	**0.1950**	**139.04**

TABLE I: **Quantitative comparison of navigation performance**. The results demonstrate that our RL-Fine-Tuned Model significantly outperforms the baselines in terms of flight speed, stability, and total exploration distance.

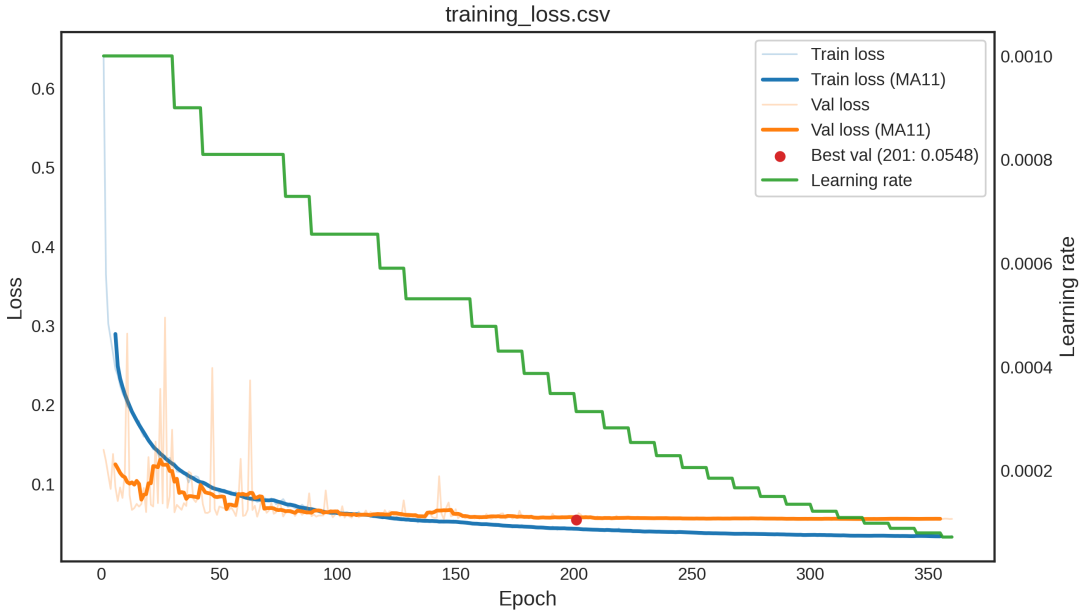

Fig. 5: This plot shows the training and validation loss curves over epochs, along with the scheduled learning rate decay. The model converges steadily with the best validation performance achieved around epoch 201, indicating stable training and limited overfitting as the learning rate decreases.

B. Quantitative Analysis of Navigation Performance

To comprehensively evaluate the effectiveness of our proposed method, we compare the Expert Policy, the Base Model, and the RL-Fine-Tuned Model using four intuitive metrics (see Table I):

- **Accident Rate**: measures the safety of the agent by tracking the frequency of collisions per second.
- **Average Speed**: reflects the navigation efficiency and the agent's aggressiveness in traversing the environment.
- **Flight Stability**: captures the variance in control commands. Improved stability is crucial for reducing hardware wear and visual jitter in real-world deployment.
- **Distance**: quantifies the total exploration capability within a fixed episode.

As shown in Table I, our RL-Fine-Tuned Model demonstrates superior performance across key navigation metrics. While the Base Model successfully mimics the conservative nature of the expert, the RL fine-tuning process significantly enhances the agent's capability. Specifically, the RL model achieves the highest Average Speed (2.314 m/s) and covers the longest Distance (139.04 m), outperforming the baseline by a large margin. Furthermore, it exhibits the best Flight Stability (0.1950), indicating that the reinforcement learning process not only improves exploration efficiency but also optimizes the smoothness of the trajectory. These results suggest that our method effectively balances high-speed exploration with stable control characteristics.

VI. Conclusion

This project presented a lightweight, end-to-end learning-based framework for quadrotor obstacle avoidance in cluttered environments. The proposed approach directly maps depth-based perception to control commands, enabling low-latency reactive navigation without explicit path planning. To address the instability ("cold-start" problem) and sample inefficiency of pure reinforcement learning, we adopted a hybrid Imitation-Reinforcement Learning strategy. A rule-based expert policy was first utilized to generate high-quality demonstrations for behavior cloning, effectively warming-start the policy. This was followed by PPO-based fine-tuning with a composite reward function that balances safety, speed, and flight smoothness.

Depth information was compressed into compact spatiotemporal occupancy grids, allowing the policy to infer obstacle geometry and motion while remaining computationally efficient

for real-time deployment. Simulation results in Gazebo/PX4 demonstrate strong generalization across randomized obstacle fields, highlighting the robustness and efficiency of the proposed method.

Nevertheless, limitations remain. Due to the end-to-end nature of the reactive controller, the generated commands may exhibit high-frequency fluctuations compared to smooth trajectory planners. Additionally, the current system is constrained to planar navigation (fixed-altitude flight). Future work will focus on extending the framework to full 3D volumetric avoidance, investigating Sim-to-Real transfer techniques for hardware deployment, and exploring recurrent architectures to handle longer-term temporal dependencies.

REFERENCES

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

[2] 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.

[3] 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.

[4] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[5] X. Fan, M. Lu, B. Xu, and P. Lu, "Flying in highly dynamic environments with end-to-end learning approach," *IEEE Robotics and Automation Letters*, 2025.

[6] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, p. eaaz9712, 2020.

[7] A. Bhattacharya, M. Cannici, N. Rao, Y. Tao, V. Kumar, N. Matni, and D. Scaramuzza, "Monocular event-based vision for obstacle avoidance with a quadrotor," *arXiv preprint arXiv:2411.03303*, 2024.

[8] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] 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.

[11] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, "Vggt: Visual geometry grounded transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5294–5306.