Mastering Diverse, Unknown, and Cluttered Tracks for Robust Vision-Based Drone Racing

Feng Yu, Yu Hu, Yang Su, Yang Deng, Linzuo Zhang, and Danping Zou

***Abstract*—Most reinforcement learning (RL)-based methods for drone racing target fixed, obstacle-free tracks, leaving the generalization to unknown, cluttered environments largely unaddressed. This challenge stems from the need to balance racing speed and collision avoidance, limited feasible space causing policy exploration trapped in local optima during training, and perceptual ambiguity between gates and obstacles in depth maps-especially when gate positions are only coarsely specified. To overcome these issues, we propose a two-phase learning framework: an initial soft-collision training phase that preserves policy exploration for high-speed flight, followed by a hard-collision refinement phase that enforces robust obstacle avoidance. An adaptive, noise-augmented curriculum with an asymmetric actor-critic architecture gradually shifts the policy's reliance from privileged gate-state information to depth-based visual input. We further impose Lipschitz constraints and integrate a track-primitive generator to enhance motion stability and cross-environment generalization. We evaluate our framework through extensive simulation and ablation studies, and validate it in real-world experiments on a computationally constrained quadrotor. The system achieves agile flight while remaining robust to gate-position errors, developing a generalizable drone racing framework with the capability to operate in diverse, partially unknown and cluttered environments.**

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

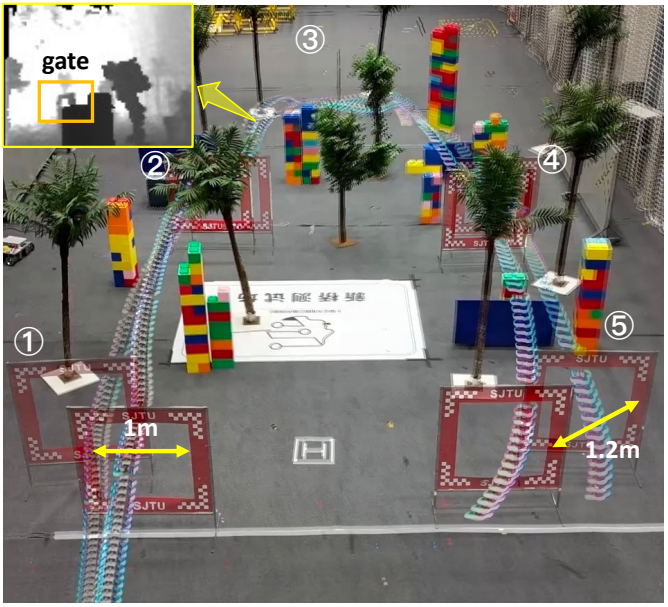

Fig. 1. Given corse prior information about gate positions and their order, our trained policy enables the quadrotor to safely traverse the unseen track with random obstacles using only onboard depth sensing and computation, demonstrating strong robustness to gate-position errors and obstacles.

I. Introduction

AUTONOMOUS drone racing represents a significant challenge in robotics, inspired by human first-person-view (FPV) racing competitions that require precise control and split-second decision-making. The task requires high-speed, collision-free navigation through a sequence of gates with minimal lap time, often aiming to surpass the performance of expert human pilots [1], [2]. Consequently, it has become a benchmark problem for evaluating the agility, environmental adaptability, and real-time decision-making capabilities of autonomous drones.

In recent years, a number of competitions have been launched to foster accelerated progress within the field, such as the IROS'16-19 Autonomous Drone Racing series [3], NeurIPS 2019's Game of Drones [4], and the 2019 AlphaPilot Challenge [5]. In these competitions, the methodology has progressively shifted from traditional perception–planning–control pipelines to end-to-end reinforcement learning. While conventional approaches suffer from cumulative errors and processing latency that limit maximum achievable speeds [6], RL-based methods significantly reduce inference time and enable direct optimization of flight performance [7]—potentially surpassing the capabilities of human pilots. However, existing RL solutions typically require extensive training on fixed, obstacle-free tracks, resulting in poor generalization and limited safety. This raises a critical research question: *Can we train a single depth-vision-based policy that generalizes across diverse, unknown tracks, adapts to noise in gate position commands, and maintains aggressive gate traversal in cluttered environments?*

Compared with standard drone racing, which is typically formalized as a time-optimal control problem in a fixed environment that pushes vehicle's physical limits, racing in unknown, cluttered environments introduces an additional requirement: the policy must jointly optimize for speed, safety, and generalization to novel scenarios. This creates an inherent trade-off between optimality and adaptability, giving rise to several key challenges. First, an exploration-exploitation conflict arises under high-speed, collision-prone settings: high-speed racing favors exploiting in-track progress, whereas obstacle-avoidance necessitates exploring collision-

Manuscript received: August 15, 2025; Revised: October 12, 2025; Accepted: November 28, 2025.

This paper was recommended for publication by Editor G. Loianno upon evaluation of the Associate Editor and Reviewers' comments.

The authors are with the Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: dpzou@sjtu.edu.cn).

Digital Object Identifier (DOI): see top of this page.

free spaces. Early termination due to collisions exacerbates this imbalance, as early terminated episodes offer limited learning signals for policy improvement. Second, gates and obstacles often share similar depth cues, such as thin structures against cluttered backgrounds, yet require opposed control responses: aggressive approach versus cautious avoidance. This perceptual ambiguity hinders the policy's ability to acquire discriminative representations without explicit semantic supervision, reducing adaptability to unfamiliar scenarios. Finally, the diversity of track layouts and obstacle configurations imposes stricter demands on the robustness and generalization capability of policy.

Contribution: This work proposes a novel RL framework for agile and robust vision-based racing. We introduce a two-phase training method that explicitly models rigid-body collisions in different manners to enhance both racing speed and obstacle avoidance. To distinguish between gates and obstacles in the depth map, we integrate an adaptive noise-augmented curriculum within an asymmetric actor-critic training framework. Simultaneously, the track primitive generator composes diverse layouts to enhance policy generalization, while Lipschitz constraints promote motion smoothness for safe flight. To accelerate convergence and reduce the sim-to-real gap, we develop DiffLab [1], a highly parallelized and high-fidelity drone simulator built on NVIDIA Isaac Lab [8]. Finally, we validate the efficacy of our framework through extensive simulations and real-world flight experiments, demonstrating robust real-world performance as shown in Fig. 1. To the best of our knowledge, this is the first vision-based autonomous drone racing system operating in diverse, unknown, and cluttered real-world environments.

II. Related Work

A. Autonomous Drone Racing without Obstacles

Optimization-based methods are predominantly designed as state-based approaches. They employ rigorous mathematical formulations to derive explicit solutions through constrained optimization frameworks. The problem is typically formulated as a minimum-time trajectory planning task with discrete waypoints constraints in obstacle-free scenarios. Foehn et al. [9] introduces a trajectory progression parameterization scheme, enabling concurrent optimization of temporal allocation and spatial trajectory design. However, this approach is computationally expensive and subsequent research attempts to address this by Model Predictive Contouring Control (MPCC) [10], which achieves real-time implementability and simultaneously maximizes path progression while minimizes tracking deviation, and its extended research [11] demonstrates inherent robustness against environmental variations and external perturbations. However, these state-based methods heavily rely on the drone and gate states without incorporating visual perception systems, thereby limiting environmental understanding and resulting in failures in obstacle-dense environments.

RL-based methods have been shown to achieve high-speed flight [12]. For example, in [13], an end-to-end racing network is trained in a simulator and the higher-level commands output by the network are used as tracking command for the MPC controller during real flight. Although the learned policy results in slower performance than optimization based methods, it shows adaptability to gate pose variations and generalization to unseen tracks to a certain extent. The state-based policy can also serve as a teacher network to distill vision-based policies [14], [15]. Notably, Kaufmann et al. [2] accurately model closed-loop systems by combining abstract representations and learned residual models, enabling the trained Swift system to outperform human pilots in real-world scenarios. Furthermore, to improve controller optimality, Ferede et al. [16] proposes a high-speed controller using end-to-end RL to directly output motor commands. In addition, Wang et al. [17] introduces an environment-shaping framework to enhance the generalization of RL agents across diverse and unseen tracks without retraining.

B. Autonomous Drone Racing in Cluttered Environments

Several optimization- and planning-based methods have demonstrated effectiveness in obstacle-rich scenarios. Penicka et al. [18] propose a sampling-based method with incrementally complex quadrotor modes to compute time-optimal trajectories, but it lacks generalization to novel environments. The winning solution of the 2022 DJI RMUA UAV Challenge [19] proposes an online replanning framework for handling dynamic gates, though its trajectory-based approach depends on accurate state estimates of both the gates and the drone.

RL-based approaches remain relatively underexplored in obstacle-aware drone racing. Penicka et al. [20] address this challenging task by using a topological path planner to guide learning the state-based racing policy. However, their policy is trained on fixed environments and does not generalize to unseen tracks. Other recent RL methods [21], [22] apply advanced techniques but lack real-world validation. In contrast, our approach trains a vision-based policy capable of zero-shot sim-to-real transfer, enabling autonomous drone racing in unseen, cluttered real-world environments.

III. Methodology

A. Overview

In this work, we tackle vision-based drone racing in diverse, unknown, and cluttered environments, where the drone navigates using noisy gate position commands and perceives the surroundings solely through depth data. This task is highly challenging due to limited field of view, ambiguous perception from depth-only sensing, command inaccuracies from coarse gate positions, and environmental noise. As illustrated in Fig. 2, our framework consists of three key components. First, we decompose the learning process into two progressive optimization stages (Sec. III-B). Second, we propose an adaptive noise-augmented curriculum learning scheme combined with an asymmetric actor–critic architecture, integrated with a track primitive generator (Sec. III-C). Finally, we introduce DiffLab, a high-fidelity, parallelized simulator (Sec. III-E). Implementation details, including system identification, domain randomization, and Lipschitz regularization are provided in Sec. III-D.

[1]https://anonymous.4open.science/r/MasterRacing-E113

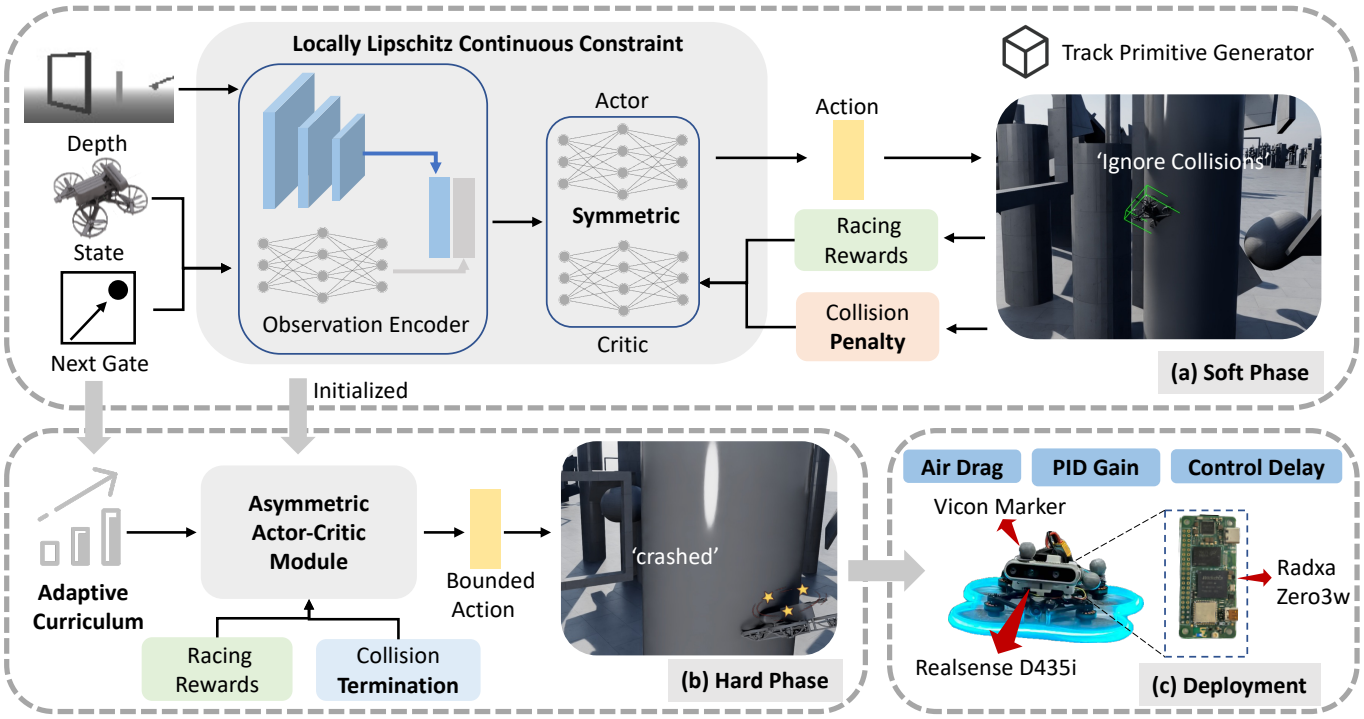

Fig. 2. The proposed two-phase framework for quadrotor racing comprises: (a) **Soft Collision Phase**: Depth observations, next gate position commands and drone states are encoded into shared embeddings for a locally Lipschitz constrained actor-critic network. Actions are executed in the collision-free simulator generated by the predefined track primitive generator to encourage racing. (b) **Hard Collision Phase**: The pre-trained policy is fine-tuned with curriculum noise injected into next gate position commands, while interacting with a rigid-body simulator enforcing real collision effects. (c) **Deployment**: After system identification aligning simulation and real dynamics, the policy is deployed on a physical quadrotor using Intel RealSense D435i for depth perception and VICON for state estimation.

B. Learning Obstacle-Aware Agile Racing

Reinforcement learning for racing in complex environments frequently encounters early termination due to rich collisions, which hinders comprehensive exploration of trial-and-error policies and breaks balance between speed optimization and obstacle avoidance. Inspired by [23], [24], we develop a two-phase reinforcement learning framework for racing which composes of staged soft or hard collision constraints. With the soft constraints, the agent can execute long trajectories to learn aggressive racing behaviors. Afterwards, it will be fine-tuned under the hard constraints to develop better avoidance ability.

$$r_{soft\_collision} = -\sum_{p} \mathbb{I}[p], \tag{1}$$

$$r_{hard\_collision} = \begin{cases} -1, \text{if collided} \\ 0, \text{else.} \end{cases} \tag{2}$$

Soft-Collision Phase. To improve racing capability, the drone is trained without rigid body collision effects in this phase, enabling it to pass through obstacles without disrupting flight or triggering early termination. This preserves trajectory continuity, facilitating more complete scene coverage for positive sample collection and racing optimization. The remaining scene configuration remains consistent with real-world conditions. To expose the drone to realistic dynamics and prevent it from over-prioritizing speed over safety, we introduce a mild collision penalty with small weight, allowing the policy to begin balancing racing and obstacle avoidance. As illustrated in Fig. 3, the penalty is computed based on the number of collision points penetrating into the environment mesh, as described in (1), where $\mathbb{I}[p] \in [0, 1]$ indicates whether the point p penetrates the environment mesh. For detection, collision points are uniformly placed on the drone body, and Warp[2] is used to check their intersections with the environment mesh in parallel.

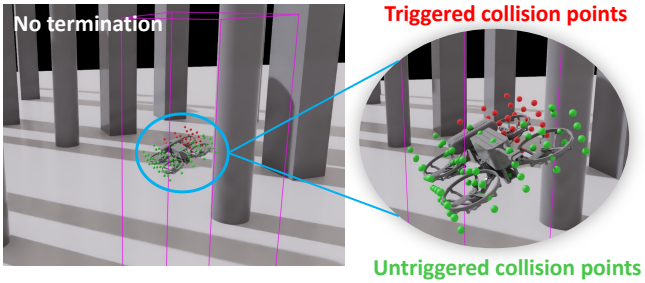

Fig. 3. The drone completes the first phase without collision termination. Collision points are uniformly placed on the collider box (green/red lattices), and a mild penalty is computed based on activated points (red).

Hard-Collision Phase. We fine-tune the pre-trained policy from the first phase in the physically realistic scenarios, where collisions can occur, triggering early termination and penalty signals as (2). This forces the drone to adhere to the scene

[2]https://nvidia.github.io/warp/

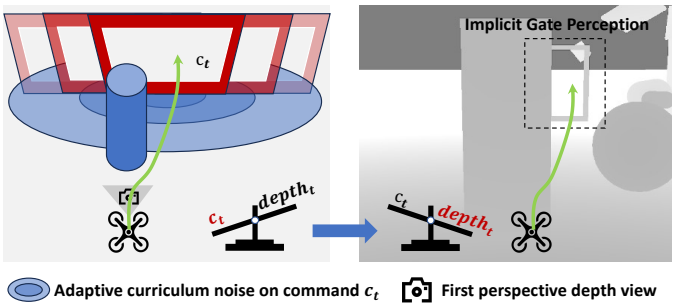

Fig. 4. The policy's reliance gradually shifts from next gate position commands to depth via the adaptive noise-augmented curriculum and asymmetric actor-critic framework.

settings and learn safer, more precise behaviors. Therefore, unlike the first phase, which primarily focuses on racing skill acquisition, this phase prioritizes exploration to better balance speed maximization and obstacle avoidance. Consequently, the weights assigned to the smoothness and collision penalties are also increased at this phase. Furthermore, an adaptive noise curriculum (see Sec. III-C) is employed in this phase to enhance the policy's ability to visually recognize gates from depth inputs.

C. Curriculum Learning for Generalizable Racing

Adaptive Noise-Augmented Curriculum. Curriculum learning is a widely used technique for acquiring complex behaviors by decomposing difficult tasks into a sequence of progressively harder subtasks [25], [26]. A common implementation involves gradually increasing the complexity of the environment, such as by raising the density of obstacles to enhance navigation robustness [27]. In this work, beyond applying curriculum learning to environment complexity to enhance obstacle avoidance capability, we further introduce a noise curriculum on the next-gate position commands.

Specifically, the adaptive noise-augmented curriculum works as follows: (1) increase the noise level if the policy performs well under the current noise, (2) decrease it if performance falls below a predefined tolerance, and (3) keep it unchanged if performance is moderate. The update rule is:

$$ {}^{i}\mu_{t+1} = {}^{i}\mu_t * (1+\alpha_1)^{\mathbb{I}[n>3]} * (1-\alpha_2)^{\mathbb{I}[n<3]} \tag{3}$$

Here, ${}^{i}\mu_t$ denotes the upper bound of the uniform distribution $\mathcal{U}(-\mu,\mu)$ controlling the single-axis position noise of the i-th agent's command during episode t. The indicator $\mathbb{I}[n>3]$ checks whether the agent successfully passes more than three gates, where n represents the number of passed gates. When $n=3$, both indicators in (3) are false, leaving the noise level unchanged and encouraging the policy to better adapt to the current noise level. Here empirical observations indicate that adopting 3 gates as the baseline yields a more rational adjustment rate for curriculum difficulty. α_1 and α_2 denote the growth and decay rates of the noise level, respectively, with $\alpha_2 > \alpha_1$ so that failed agents are reset with less noisy commands for easier relearning. Because noise adjustment is performance-driven, it avoids the performance degradation that would occur from directly introducing excessive noise from scratch, maintaining a balance between noise intensity and racing performance.

Given the noise introduced by curriculum, we integrate the framework with an asymmetric actor-critic setup [28]. Specifically, as noise gradually increases, these command observations become unreliable, compelling the actor to implicitly extract gate-relevant features from depth rather than noisy command observations for generating reward-maximizing actions, while the critic still uses the ground-truth commands for precise and robust training. This combination enables a gradual shift in reliance—from external command guidance to the agent's own learned perception, thereby promoting policy robustness and enabling flight behaviors that are aware of gate structures in the depth map, as illustrated in Fig. 4.

Track Primitive Generator. Although the adaptive noise-augmented curriculum initially facilitates the policy's adaptation to various gate position inputs, further adaptation to diverse track configurations remains essential. Based on prior observations of typical motion primitives in drone racing tracks, we design three types of training tracks: circular, zigzag, and elliptical. The circular track includes both clockwise and counterclockwise directions, implicitly capturing left and right turning behaviors. The zigzag track extends this by incorporating straight-line flight. The elliptical track integrates the motion patterns of both circular and zigzag tracks and further requires autonomous switching between them. Consequently, the policies trained on these three track types acquire transferable flight proficiencies—including straight-line flight, left turns, and right turns—that enable effective racing on most tracks.

D. Policy Training

1) Observation: The policy observations $\boldsymbol{o}_t$ at time t consists of:

$$\boldsymbol{o}_t = [{}^{b}\boldsymbol{v}_t, \mathbf{r_3}, {}^{b}\boldsymbol{\delta p}_t^1, {}^{b}\boldsymbol{\delta p}_t^2, \boldsymbol{a}_{t-1}, \boldsymbol{I}_t] \tag{4}$$

Here ${}^{b}\boldsymbol{v}_t$ denotes the linear base velocity in the body frame, and $\mathbf{r}_3$ represents the third row of the rotation matrix, encoding roll and pitch angles. The command ${}^{b}\boldsymbol{\delta p}_t^1$ and ${}^{b}\boldsymbol{\delta p}_t^2$ represent the relative translation from the drone to the next gate, and from the next gate to the second subsequent gate, respectively, both expressed in the body frame. Both translation vectors are injected with noise, as described in III-C. The previous action $\boldsymbol{a}_{t-1}$ is included to encourage smooth control, making the policy aware of past commands rather than purely reactive. Following [29], which employs a low-resolution depth image for obstacle avoidance, we downsample the depth image $\boldsymbol{I}_t$ to 96×72. This reduces observation dimensionality, improving sample efficiency, and retaining sufficient gate-relevant information.

2) Action: The policy outputs a control command at each timestep t, comprising mass-normalized collective thrust and body rates (CTBR) following [30].

3) Reward: Most reward functions follow [2], with minor modifications for the progress and perception rewards. The progress reward directs the drone toward the target, while the perception reward encourages it to face the target, enhancing

TABLE I
REWARD FUNCTION TERMS

Reward Term	Expression	Weight
Towards Reward	$\frac{(\boldsymbol{p}_w^{gate}-\boldsymbol{p}_{wb})\cdot\boldsymbol{v}_w}{\|\boldsymbol{p}_w^{gate}-\boldsymbol{p}_{wb}\|\cdot\|\boldsymbol{v}_w\|}$	1
Body Rate Penalty	$\|\boldsymbol{\omega}_b\|$	$-0.02, -0.1^*$
Action Rate Penalty	$\|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|^2$	$-0.01, -0.05^*$
Collision Penalty	Eq. (1)	50
Perception Reward	Eq. (2)* $Normalize(\boldsymbol{q}_t^{-1} \odot (\boldsymbol{p}_w^{gate} - \boldsymbol{p}_{wb})) \cdot [1,0,0]^T$	100^* 0.1
Success Reward	$\mathbb{I}[\|\boldsymbol{p}_w^{gate} - \boldsymbol{p}_{wb}\| < 0.35] \cdot \frac{1}{1+\|\boldsymbol{p}_w^{gate}-\boldsymbol{p}_{wb}\|^2}$	$10, 20^*$
Bad Pose Penalty*	$\mathbb{I}[\|roll\| > \frac{\pi}{2}] \cdot \mathbb{I}[\|pitch\| > \frac{\pi}{2}]^*$	-30^*

$*$ denotes the settings for the second phase.

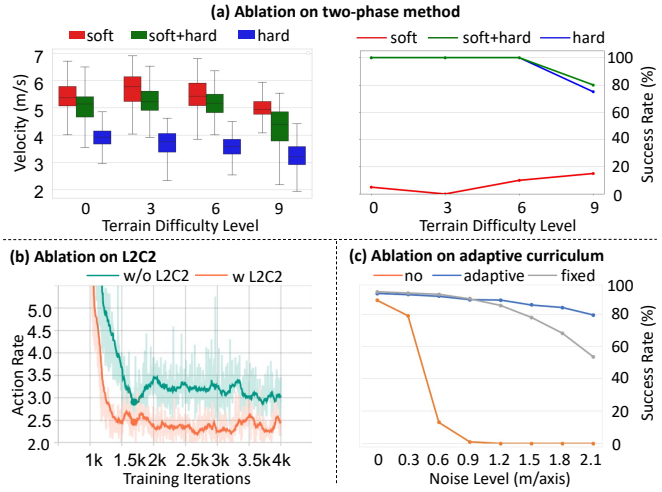

Fig. 5. **Ablation results.** (a) Comparison of velocity and success rate across training phases: soft-collision only (red), combined soft- and hard-collision (green), and the baseline hard-collision-only method (blue). (b) Action rate curves showing policy behavior with and without L2C2 regularization. (c) Success rate comparison under three noise curricula: no noise (orange), fixed noise (gray), and our adaptive noise curriculum (blue).

safety and task success. The reward function is largely consistent across the two-phase training, with only slight adjustments to component weights (Table I).

4) Network Design: Our network architecture is shown in Fig. 2. An MLP is used to process the state-only and command observations, while a CNN processes the depth images. The resulting feature vectors are concatenated and fed into subsequent MLP layers, which finally output CTBR control commands or value estimates.

5) Gate Crossing Detector: Because noise in the relative gate positions makes position-based gate-crossing checks unreliable, we design an automatic gate-crossing detector based on network embeddings. Specifically, we collect a balanced set of positive and negative samples in the simulator using a well-trained policy. Each sample consists of a pair: network embeddings and a corresponding gate-crossing signal. The prediction head is then trained offline using those samples with the rest of the policy network frozen. This head remains excluded from updates during policy training.

6) Sim-to-real Transfer: To narrow the sim-to-real gap and enhance policy robustness, we apply extensive domain randomization to key drone parameters, including air drag coefficients, mass, inertia, and the PID gains of the angular velocity controller. In addition to the adaptive command noise, Gaussian noise is injected into other observation channels. Following [31], [32], [33], we adopt local Lipschitz continuity constraint (L2C2) to suppress action oscillations. This method regularizes both the actor and critic to enforce local Lipschitz continuity, enhancing temporal smoothness of the action and value functions. Specifically, an additional loss term is introduced to bound the output variation between consecutive states:

$$\mathcal{L}_{L2C2} = \lambda_1 D(\pi_\theta(o_t), \pi_\theta(\hat{o}_t)) + \lambda_2 \|V_\phi(o_t) - V_\phi(\hat{o}_t)\|_2^2, \tag{5}$$

$$\hat{o}_t = o_t + (o_{t+1} - o_t) \cdot u, u \sim U(-1, 1) \tag{6}$$

where λ_1 and λ_2 are weighting coefficients, D represents the squared Hellinger distance, π_θ and V_ϕ are the policy and value networks parameterized by θ and ϕ, respectively, and $\hat{o}_t$ is the interpolated observations between o_t and o_{t+1}.

E. Quadrotor Simulator

According to quadrotor dynamics and widely-used control methods [34], [35] , we develop DiffLab based on Isaac Lab. Compared with previous work, we focus more on the sim-to-real gap. Specifically, we first adopt a more accurate air drag model that includes both first-order and second-order terms. We then account for control delay to better capture the angular velocity response. All parameters are calibrated using real flight data. Similar to AirGym [36], we provide multiple control levels, including position commands and yaw (PY), linear velocity commands and yaw (LV), CTBR and single-rotor thrusts (SRT). These controllers operate in a cascade PID loop, and the results are ultimately converted into total thrust and torque acting on the rigid body of the quadrotor.

IV. EXPERIMENTS

We employ the Proximal Policy Optimization (PPO) algorithm [37] to train the policy in DiffLab. Training on a standard computer with an Intel i7 CPU and an Nvidia RTX 4090 GPU, the policy converges in approximately 4 hours.

A. Ablation Studies

This section provides an analysis of the design of each module of our algorithm. We measure the effectiveness using three metrics: average speed, success rate, and action rate, where success rate refers to the proportion of 20 drones completing the track without collision within 8 seconds.

- **Two-Phase vs One-Phase**: We compare the performances of policies trained via two-phase and one-phase methods, with all settings maintained consistently between these approaches. Fig. 5 (a) illustrates that the two-phase method achieves higher average speed across

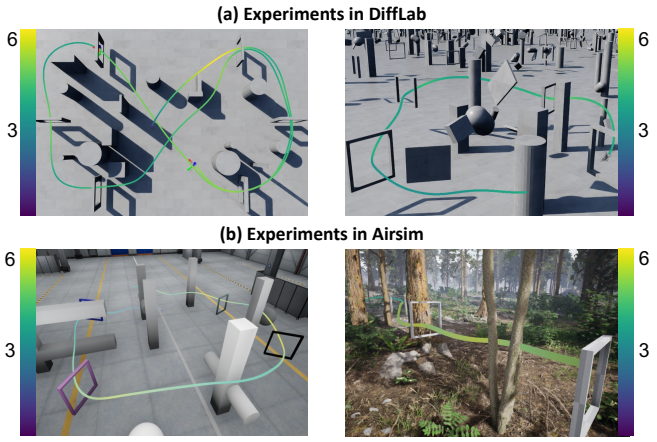

Fig. 6. **Simulation results**: We evaluate the policy in both DiffLab and Airsim simulator on different tracks with different obstacle layouts.

terrains of all difficulty levels[3], while its second phase maintains the similar success rate as the one-phase method. These results demonstrate that the two-phase method can exploit racing ability gained in the first phase while exploring and improving obstacle avoidance capability in the second phase. In contrast, the one-phase policy is more conservative and unable to fully exploit its racing potential.

- **Adaptive Noise-Augmented Curriculum**: We validate the effectiveness of our adaptive noise-augmented curriculum via comparison with noise-free and fixed-noise (1 m/axis) baselines. Results in Fig. 5 (c) demonstrate that the curriculum significantly enhances the policy robustness under various level of noise. For the noise-free command policy, performance degrades notably as noise increases, with success rate approaching 0 at 0.9 m noise. Similarly, the fixed-noise policy shows severe degradation under high noise, achieving only 54% success rate at 2.1 m noise. In contrast, our method achieves nearly 80% success rate under the same conditions.
- **Locally Lipschitz Continuous Constraint**: We compare the action rates of the policies trained with and without L2C2, as shown in the Fig. 5 (b). It can be seen that using L2C2 can significantly reduce the action rate, decreasing motion oscillation by 25% and resulting in smoother actions.

B. Simulation Experiments

We conduct extensive experiments in both our DiffLab simulator and the AirSim simulator [38]. In DiffLab, we employ a structured environment similar to the training setup, while in AirSim, we evaluate the policy in factory and forest environments. As illustrated in Fig. 6, multi-perspective visualizations of the flight process demonstrate that our policy generalizes effectiely to unseen, realistic environments and novel track layouts, achieving peak speeds at 5 m/s in both forest and factory scenes. This result confirms the effectiveness of the track primitive generator, enhancing the policy's generalization capability to diverse tracks and scenarios.

[3]Difficulty is determined by obstacle density, track layout, and gate size; a few obstacles remain even at level 0.

TABLE II
SIM-TO-SIM PERFORMANCE

Range (m/axis)	Factory		Forest	
	SR	Max Vel (m/s)	SR	Max Vel (m/s)
$[0.0, 0.0]$	10/10	5.1	10/10	5.3
$[-0.3, 0.3]$	10/10	4.9	10/10	5.1
$[-0.6, 0.6]$	10/10	4.4	10/10	4.6
$[-0.9, 0.9]$	7/10	4.4	8/10	4.6
$[-1.2, 1.2]$	5/10	4.3	8/10	4.6

SR and Max Vel denote success rate and maximum velocity, respectively.

We further evaluate the policy's robustness to gate position noise in simulation to demonstrate its gate perception-aware capability. As shown in Table II,the policy achieves a 100% success rate across all scenarios when the single-axis position error is below 0.6 m. Even with errors approaching 1.2 m, it maintains a success rate above 50%. Moreover, the maximum speeds exceed 5 m/s in both test environments, with only moderate performance degradation as noise increases. These results validate the effectiveness of our adaptive noise-augmented curriculum in enhancing the policy's robustness to gate position errors. Notably, the gradual decline in success rate under high-noise conditions is not due to collisions but rather to incorrect commands caused by excessive noise. These misguiding 'next-gate position' commands lead the drone to deviate significantly from the track and fail to cross gates. While this results in task failure, it demonstrates the policy's strong obstacle avoidance capabilities.

C. Real World Experiments

We validate our policy on a lightweight quadrotor platform (0.46 kg), equipped with an Intel Realsense D435i depth camera and a Radxa Zero3W onboard computer (1.6GHz quad-core A55 CPU with 1TOPS NPU)[4] as illustrated in Fig.2 (c). A high-precision motion capture system running at 100Hz provides accurate drone motions, while low-level control is executed via a Betaflight controller.

We conduct experiments in various track and obstacle layouts, including a zigzag track, a circular track and a U-shaped track, as shown in Fig. 7. The quadrotor successfully navigates through all gates in cluttered environments, achieving a maximum speed exceeding 5 m/s, as illustrated in Fig. 8. Additionally, accurate gate-crossing detection results are also presented in Fig. 8. These results demonstrate the robustness and generalization capabilities of our method in real-world scenarios.

To further validate our policy's robustness to gate position errors, we conducted two sets of experiments: one on a zigzag track and the other on a U-shaped track. For each track, we first measured the gate positions and used them as commands for an initial flight. Then, we introduced random perturbations

[4]https://radxa.com/products/zeros/zero3w

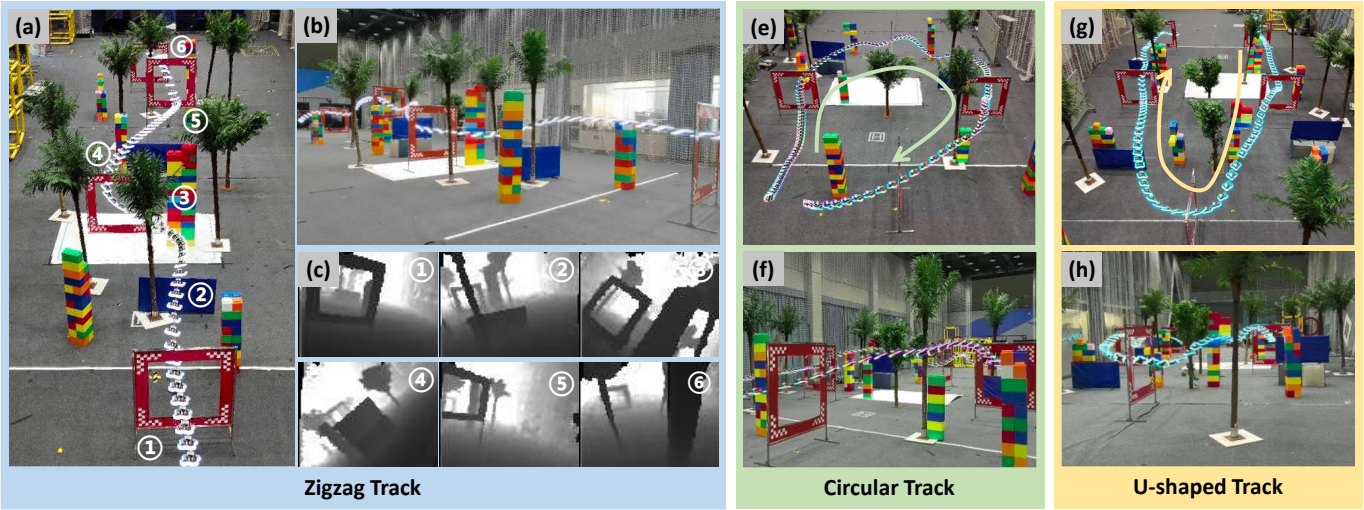

Fig. 7. **Real-world results.** We validate our framework in various unknown, cluttered environments, including (a) a zigzag track, (d) a circular track, and (f) a U-shaped track with randomly placed obstacles such as trees and boxes. The quadrotor's flight trajectory is visualized by overlaying its position in the video footage (please refer to the supplementary video for more details). (b), (e), and (g) show side views of the trajectories, while (c) presents the front depth view corresponding to (a).

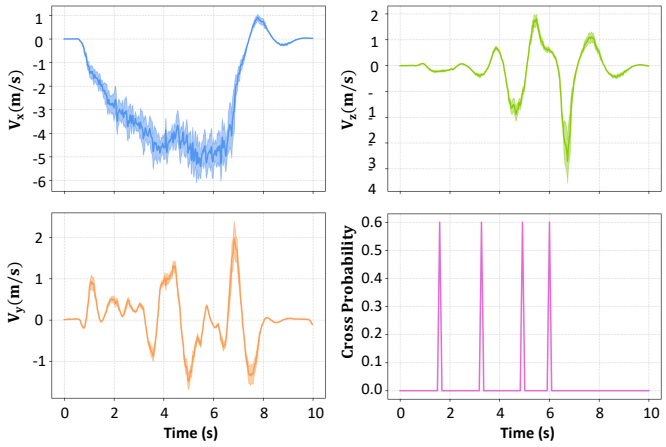

Fig. 8. **Velocity and Gate Detection Results**. We present the velocity measurements along all three axes, recorded by the motion capture system, as well as the gate detection results output by our policy during flight on the zigzag track shown in Fig. 7 (a).

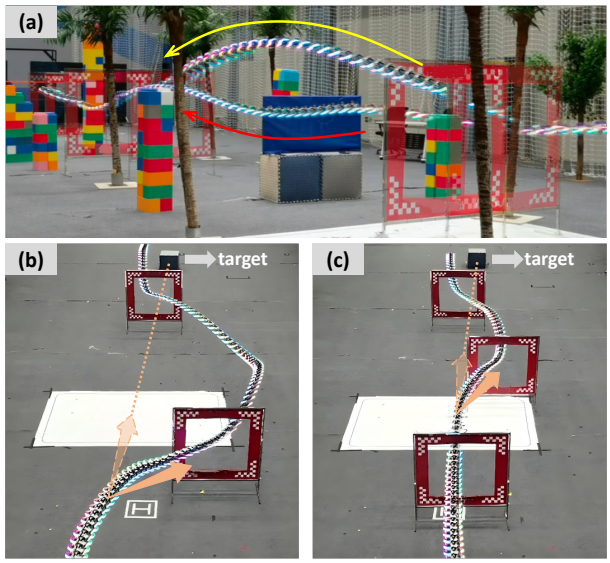

Fig. 9. **Robustness Tests.** (a) Gate positions are randomly perturbed while keeping the gate position commands fixed. Trajectories from the perturbed track (second run) are overlaid on the initial flight for comparison. (b, c) The target command is set beyond the track's end. Despite inaccurate gate commands, the drone demonstrates gate perception-aware behavior by autonomously adjusting its flight to pass through the gates.

to the gate positions and performed a second flight using the original commands. Results shown in the Fig. 1 and Fig. 9 (a) demonstrate that our policy tolerate noise exceeding 1 meter, flying swiftly and accurately through all gates. This highlights the policy's gate perception-aware capability to correct noisy commands. We further validated this in real-world tests, as shown in Fig. 9 (b) and Fig. 9 (c). The target command is set directly at the end of the track, keeping the gate within the camera's view but misaligned with the command direction. Despite this, the drone adjusted its trajectory to pass through the gate instead of flying straight, confirming the policy's perception-driven robustness to gate position errors.

V. Conclusion And Limitation

In this work, we propose a learning-based approach for vision-based drone racing, enabling generalization from fixed obstacle-free tracks to diverse tracks in cluttered environments. Our method combines two-phase training with an adaptive noise-augmented curriculum to develop robust gate perception-aware racing capability. Extensive experiments validate the effectiveness of the proposed framework.

Limitations: While our approach generalizes well across diverse racing scenarios, there is still room for improvement in the flight speed of drones, and the distribution of scenes will greatly affect the speed of drones. It is possible to further study the balance between optimality and safety in various scenarios, as well as perception-based pose correction to reduce reliance

on external localization.

REFERENCES

[1] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," *IEEE Transactions on Robotics*, 2024.

[2] 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.

[3] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. Wagter, G. Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T.-C. Au, and S. J. Kim, "Challenges and implemented technologies used in autonomous drone racing," *Intell. Serv. Robot.*, vol. 12, p. 137–148, Apr. 2019.

[4] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," in *Neurips 2019 competition and demonstration track*, pp. 177–191, PMLR, 2020.

[5] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[6] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, 2019.

[7] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

[8] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[9] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[10] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.

[11] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022.

[12] K. Nagami and M. Schwager, "Hjb-rl: Initializing reinforcement learning with optimal control policies applied to autonomous drone racing.," in *Robotics: science and systems*, pp. 1–9, 2021.

[13] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1205–1212, IEEE, 2021.

[14] J. Fu, Y. Song, Y. Wu, F. Yu, and D. Scaramuzza, "Learning deep sensorimotor policies for vision-based autonomous drone racing," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5243–5250, IEEE, 2023.

[15] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," *arXiv preprint arXiv:2403.12203*, 2024.

[16] R. Ferede, C. De Wagter, D. Izzo, and G. C. De Croon, "End-to-end reinforcement learning for time-optimal quadcopter flight," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6172–6177, IEEE, 2024.

[17] H. Wang, J. Xing, N. Messikommer, and D. Scaramuzza, "Environment as policy: Learning to race in unseen tracks," *arXiv preprint arXiv:2410.22308*, 2024.

[18] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[19] Q. Wang, D. Wang, C. Xu, A. Gao, and F. Gao, "Polynomial-based online planning for autonomous drone racing in dynamic environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1078–1085, IEEE, 2023.

[20] 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.

[21] Y. Liu, "Learning generalizable policy for obstacle-aware autonomous drone racing," *arXiv preprint arXiv:2411.04246*, 2024.

[22] W. Xiao, Z. Feng, Z. Zhou, J. Sun, G. Wang, and J. Chen, "Time-optimal flight in cluttered environments via safe reinforcement learning," *arXiv preprint arXiv:2406.19646*, 2024.

[23] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," *arXiv preprint arXiv:2309.05665*, 2023.

[24] H. Wang, Z. Wang, J. Ren, Q. Ben, T. Huang, W. Zhang, and J. Pang, "Beamdojo: Learning agile humanoid locomotion on sparse footholds," *arXiv preprint arXiv:2502.10363*, 2025.

[25] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[26] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter, "Learning agile locomotion on risky terrains," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11864–11871, IEEE, 2024.

[27] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "Navrl: Learning safe flight in dynamic environments," *IEEE Robotics and Automation Letters*, 2025.

[28] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.

[29] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, "Learning vision-based agile flight via differentiable physics," *Nature Machine Intelligence*, pp. 1–13, 2025.

[30] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 10504–10510, IEEE, 2022.

[31] Z. Yin, C. Zheng, S. Guo, Z. Wang, and S. Zhao, "Taco: General acrobatic flight control via target-and-command-oriented reinforcement learning," *arXiv preprint arXiv:2503.01125*, 2025.

[32] Y. Zhang, B. Nie, and Y. Gao, "Robust locomotion policy with adaptive lipschitz constraint for legged robots," *IEEE Robotics and Automation Letters*, 2024.

[33] T. Kobayashi, "L2c2: Locally lipschitz continuous constraint towards stable and smooth reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4032–4039, IEEE, 2022.

[34] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[35] T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor uav using geometric methods on se (3)," *arXiv preprint arXiv:1003.2005*, 2010.

[36] K. Huang, H. Wang, Y. Luo, J. Chen, J. Chen, X. Zhang, X. Ji, and H. Liu, "A general infrastructure and workflow for quadrotor deep reinforcement learning and reality deployment," *arXiv preprint arXiv:2504.15129*, 2025.

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

[38] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*, 2017.