2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Curriculum Reinforcement Learning for Quadrotor Racing with Random Obstacles

Fangyu Sun, Fanxing Li, Yu Hu, Linzuo Zhang, Yueqian Liu, Wenxian Yu*, Danping Zou*

***Abstract*— Autonomous drone racing has attracted increasing interest as a research topic for exploring the limits of agile flight. However, existing studies primarily focus on obstacle-free racetracks, while the perception and dynamic challenges introduced by obstacles remain underexplored, often resulting in low success rates and limited robustness in real-world flight. To this end, we propose a novel vision-based curriculum reinforcement learning framework for training a robust controller capable of addressing unseen obstacles in drone racing. We combine multi-stage cu rriculum learning, domain randomization, and a multi-scene updating strategy to address the conflicting challenges of obstacle avoidance and gate traversal. Our end-to-end control policy is implemented as a single network, allowing high-speed flight of quadrotors in environments with variable obstacles. Both hardware-in-the-loop and real-world experiments demonstrate that our method achieves faster lap times and higher success rates than existing approaches, effectively advancing drone racing in obstacle-rich environments. The video and code are available at: `https://github.com/SJTU-ViSYS-team/CRL-Drone-Racing`.**

I. INTRODUCTION

Vision-based autonomous drone racing in cluttered environments represents a significant challenge, requiring vehicles to fly as fast as possible while avoiding obstacles. Although the remarkable agility of autonomous quadrotors has been demonstrated in controlled laboratory settings [1], achieving collision-free racing in densely cluttered environments remains an unresolved problem. By definition, obstacle-aware racing necessitates that the drone maximizes its speed while avoiding obstacles, pushing the platform's dynamics to its operational limits. However, since the gates are also perceived as obstacles, the dual objectives of gate passing and obstacle avoidance are inherently conflicting, posing a significant challenge for both tasks. These competing demands, combined with environmental variability and model mismatches, can lead to catastrophic failures, highlighting the need for robust and adaptive solutions.

Recent efforts have sought to integrate obstacle avoidance into drone racing through several methodologies. Traditional path-planning and optimization-based techniques enable high-speed flight and effectively trade off lap time against computational cost [2]–[4]. However, their performance heavily depends on careful algorithm design and can suffer from model mismatches. In contrast, learning-based approaches leverage reinforcement learning (RL) [5], [6] and imitation learning (IL) [7] to train policies that achieve low-latency, collision-free control. However, these policies struggle to generalize across racetracks and obstacle configurations, and often fail to transfer reliably from simulation to the real world due to the low success rate.

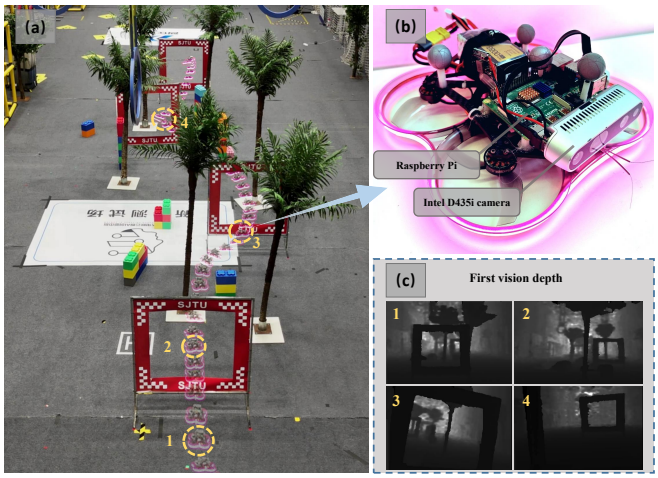

Fig. 1: **Our quadrotor autonomously races with random obstacles in the real world at speeds of up to 8m/s.** (a) The real-world experiment of the S-shaped racetrack and the trajectory. (b) Our onboard drone is equipped with a Raspberry Pi computer and an Intel D435i depth camera. (c) FPV depth images during racing with obstacles.

To address these limitations, we propose a curriculum RL framework for training an end-to-end vision-based control policy, enabling high-speed quadrotor flight in unseen cluttered environments. Our framework employs a multi-stage curriculum learning, allowing the agent to progress from simple obstacle-free navigation to complex racing in cluttered racetracks.

During training, an obstacle generator with safe margins is incorporated, along with randomized initial states, to facilitate efficient exploration of obstacle-rich environments. To enhance training efficiency, we adapt a customized multi-scene updating mechanism that dynamically adjusts the number of training scenes based on curriculum difficulty, thereby addressing the generalization challenge of policies across diverse obstacle configurations.

To resolve the inherent conflict between gate traversal and obstacle avoidance, we devise a well-structured reward function that strikes a balance between collision avoidance penalty and gate-passing reward. This reward design also

This work was supported by the National Key Research and Development Program of China (2022YFB3903801) and the National Science Foundation of China (62073214).

Fangyu Sun, Fanxing Li, Yu Hu, Linzuo Zhang, Wenxian Yu, and Danping Zou are with the School of Automation and Perception, Shanghai Jiao Tong University, China (e-mail: dpzou@sjtu.edu.cn).(*) denotes the corresponding author.

Yueqian Liu is with the Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands.

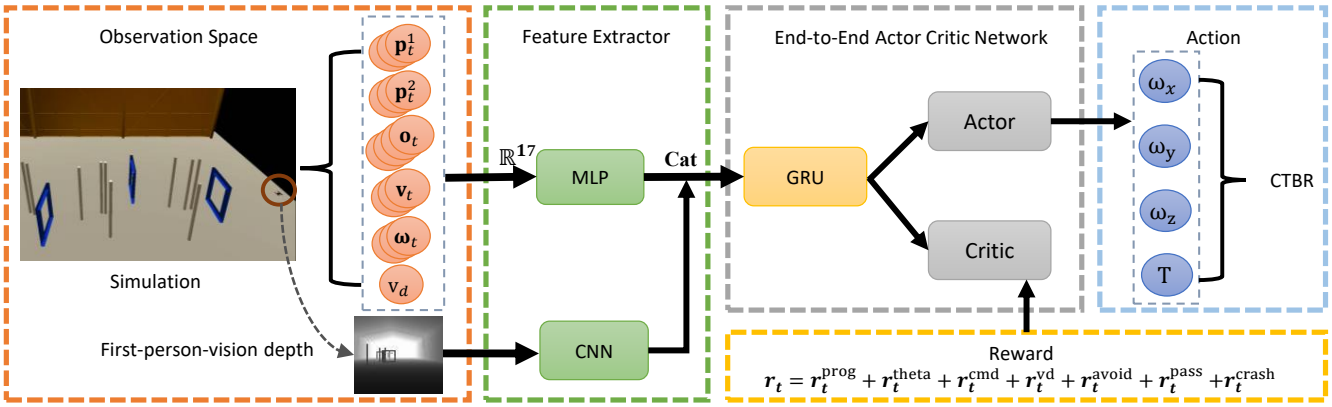

Fig. 2: **The framework of our RL policy training network for the obstacle-rich racing task.** The network architecture consists solely of simple CNNs and MLPs without any complex backbones. "Cat" is the abbreviation for "Concatenate".

enables rapid yaw maneuvers during high-speed flight, enhancing obstacle awareness while preserving racing performance. Our network architecture is lightweight, which supports real-time onboard inference on a Raspberry Pi. Our main contributions are summarized as follows:

- **Curriculum learning for systematic generalization.** We propose a multi-stage curriculum learning strategy that progressively increases task difficulty through an obstacle generator and random initialization. This addresses the limited generalization of prior methods [5], [6] by enabling the agent to learn robust policies that transfer across unseen cluttered environments.
- **Multi-scene training for improved efficiency.** We introduce a novel multi-scene training paradigm that enhances training efficiency in RL while, more importantly, enabling consistent policy generalization across diverse obstacle configurations.
- **Reward design resolving the conflicting problem.** We design effective reward functions that resolve the inherent conflict between obstacle avoidance and gate passing, allowing the agent to perform high-speed flight with large-angle perception maneuvers without compromising safety.

Experimental results demonstrate that our approach consistently outperforms prior methods in both success rate and lap time. Validated through hardware-in-the-loop and real-world flights across three distinct racetracks, our method achieves speeds up to $10\,m/s$ and $8\,m/s$ respectively, with 100% success rates in all obstacle-rich environments.

II. Related Work

A. Learning Obstacle-Avoidance Flight

In recent years, learning-based methods have been increasingly applied to obstacle avoidance for autonomous drones. Loquercio et al. [8] show that an imitation learning (IL) approach can train a neural network entirely in simulation by imitating an expert with full environment knowledge. The resulting policy directly maps noisy onboard sensor observations to collision-free trajectories. Yu et al. [9] introduce an RL-based pipeline utilizing depth map inputs, enabling the drone to dynamically adjust its flight speed according to environmental complexity without collision. Similarly, employing RL, Zhao et al. [10] propose a hierarchical learning and planning framework that dynamically adjusts the speed constraints of a model-based trajectory planner. Zhang et al. [11] leverage the differentiable quadrotor dynamics to learn obstacle-avoidance policy through direct gradient backpropagation. Building on this work, Hu et al. [12] replace the depth map with optical flow to achieve obstacle avoidance with a monocular FPV setup.

However, those methods only consider obstacle avoidance and neglect the requirement of gate traversal. Importantly, these two objectives are inherently conflicting: avoidance encourages conservative maneuvers to steer clear of obstacles, whereas gate traversal demands aggressive and precise flight, often at proximity to the gate center, which itself is perceived as an obstacle.

B. Autonomous Drone Racing

Autonomous drone racing has emerged as a growing area of research where drones are required to complete a predefined sequence of waypoints in minimal time. Success in this domain necessitates the integration of sophisticated hardware and complex algorithms capable of perceiving the environment, planning optimal paths, and executing actions in real-time. Song et al. [13] indicate that RL controllers can outperform traditional optimization-based methods, such as those proposed in [14] and [15]. The emergence of this advantage stems from RL's ability to effectively handle highly nonlinear dynamical systems and complex objectives, which pose significant challenges to traditional optimization methods like model predictive control (MPC) [16], [17]. Recent research [18] has shown that policies trained with deep RL can even surpass human world champions. Furthermore, recent vision-based drone racing research has demonstrated the ability to bypass traditional state estimation entirely by training an asymmetric actor-critic network to map visual inputs directly to control commands [19].

Despite the strong performance of RL-based methods in obstacle-free racing scenarios, their application to environ-

ments with obstacles remains relatively underexplored. To date, only a limited number of studies [5], [6] have addressed this challenging problem. However, these methods either overfit to a single predefined obstacle-aware racing track or underfit due to excessive domain randomization, resulting in low success rates and poor transfer to unseen real-world scenes. Motivated by this gap, we aim to develop a robust policy that adapts to varying obstacle environments while maintaining both high flight speed and success rate.

III. Methodology

A. Dynamics and Task Formulation

We consider a quadrotor with mass m and diagonal inertia matrix $\mathbf{J}$. Its dynamics are governed by

$$\dot{\mathbf{p}}_W = \mathbf{v}_W, \qquad \dot{\mathbf{v}}_W = \frac{1}{m}\mathbf{R}_{WB}(\boldsymbol{f}_T + \boldsymbol{f}_D) + \boldsymbol{g}$$
$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q}\hat{\boldsymbol{\Omega}}, \qquad \dot{\boldsymbol{\omega}} = \mathbf{J}^{-1}(-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} + \boldsymbol{\tau}_T + \boldsymbol{\tau}_D) \quad (1)$$

where $\mathbf{p}_W$ and $\mathbf{v}_W$ denote position and velocity in the world frame, $\mathbf{R}_{WB}$ is the rotation matrix from body to world frame, and $\boldsymbol{\omega}$ is the angular velocity expressed in the body frame. The term $\hat{\boldsymbol{\Omega}}$ represents the skew-symmetric matrix of $\boldsymbol{\omega}$, while $\boldsymbol{g}$ is the gravity vector. The collective thrust along the body z-axis and the body torque generated by the four rotors are denoted by f_T and $\boldsymbol{\tau}_T$, respectively. To capture the dynamics of aggressive flight, we incorporate air drag effects through the force and torque terms $\boldsymbol{f}_D$ and $\boldsymbol{\tau}_D$.

The obstacle-rich racing task is framed as an infinite-horizon Markov Decision Process (MDP), characterized by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ are continuous. Here, $\mathcal{P}$ denotes the transition probability, $\mathcal{R}$ is the reward function, and γ is the discount factor. The objective is to find an optimal policy π^* that maximizes the expected discounted cumulative reward, formalized as

$$\pi_\theta^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]. \quad (2)$$

In the subsequent sections, we present our vision-based RL controller and the associated policy learning methodology.

B. End-to-End Vision-based Controller

An overview of our method is given in Fig. 2. Our approach directly maps vision-based observations to collective thrust and body rate (CTBR). In the following sections, we introduce key components of our end-to-end controller, including observation and action spaces, reward functions, and the network architecture.

1) Observation Space: The observation space consists of drone's state $\mathbf{s}^{\text{drone}}$ and depth map $\mathbf{s}^{\text{depth}}$. We define the state-based observation as

$$\mathbf{s}_t^{\text{drone}} = \left[\mathbf{p}_t^1, \mathbf{p}_t^2, \mathbf{v}_t, v_d, \mathbf{o}_t, \boldsymbol{\omega}_t\right] \in \mathbb{R}^{17}. \quad (3)$$

The $\mathbf{p}_t^1 \in \mathbb{R}^3$ and $\mathbf{p}_t^2 \in \mathbb{R}^3$ correspond to the relative position of the drone to the first and second gate's center. The $v_d \in \mathbb{R}^1$ is the desired speed, which represents the norm of the velocities along the three axes. The $\mathbf{v}_t \in \mathbb{R}^3$, $\mathbf{o}_t \in \mathbb{R}^4$ and $\boldsymbol{\omega}_t \in \mathbb{R}^3$ represent the drone's linear velocity, orientation, and angular velocity, respectively. The relative position, linear velocity, and orientation are all transferred to the body axis. Our obstacle-rich racing task also requires additional visual input to guide gate passing and obstacle avoidance. The depth map $\mathbf{s}_t^{\text{depth}} \in \mathbb{R}^{64\times64}$ from a depth camera is used. To reduce the sim-to-real gap caused by vision input, we take the inverse depth and add random Gaussian noise during training. So the whole observation space is $O_t = (\mathbf{s}_t^{\text{drone}}, \mathbf{s}_t^{\text{depth}})$.

2) Action Space: The policy is trained to directly map the observation to CTBR commands, defining the action as $\boldsymbol{u}_t = [T, \omega_x, \omega_y, \omega_z] \in \mathbb{R}^4$. The first dimension represents the mass-normalized collective thrust, while the last three dimensions represent the angular velocities.

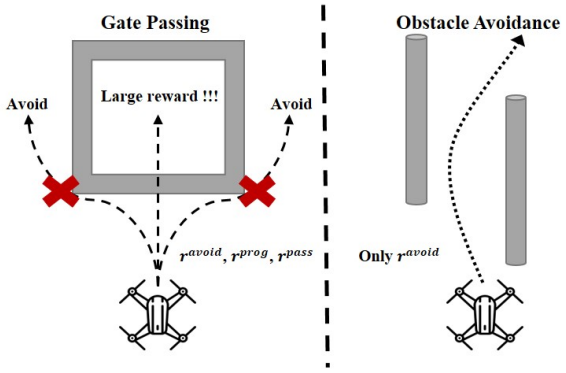

Fig. 3: **Balancing the tasks of gate passing and obstacle avoidance is challenging for training an RL policy.** These two tasks are contradictory for reward design, the obstacle-avoidance reward would cause the drone to bypass the gates from the sides rather than passing through them directly.

3) Reward Functions: Unlike previous work that focused solely on crossing gates [19], [20], our policy needs to introduce additional rewards to avoid obstacles. Moreover, as shown in Fig. 3, since the gates are also perceived as obstacles, the obstacle-avoidance reward would cause the drone to bypass the gate from the sides when passing through it. Therefore, we need to design reward functions that balance the conflicting tasks of gate passing and obstacle avoidance for the end-to-end control policy.

We employ a composite reward to guide policy learning for racing in cluttered environments. At each timestep, the overall reward R_t is

$$R_t = r_t^{\text{prog}} + r_t^{\text{theta}} + r_t^{\text{cmd}} + r_t^{\text{vd}} + r_t^{\text{avoid}} + r_t^{\text{pass}} + r_t^{\text{crash}}, \quad (4)$$

where r_t^{prog} encourages progress towards the next gate to be passed [13]. r_t^{theta} is responsible for aligning the drone to the next gate and for seeing the obstacles. Meanwhile, r_t^{theta} is also the key to enabling the agent to perform large-angle maneuvers based on the gate's pose. r_t^{cmd} is used to penalize large actions, making the trajectory smoother, r^{vd} penalizes high speed that exceeds the desired velocity. Unlike previous work [5] that relied solely on the discrete reward to achieve obstacle avoidance through trial-and-error during training, we design a simple continuous reward r_t^{avoid}

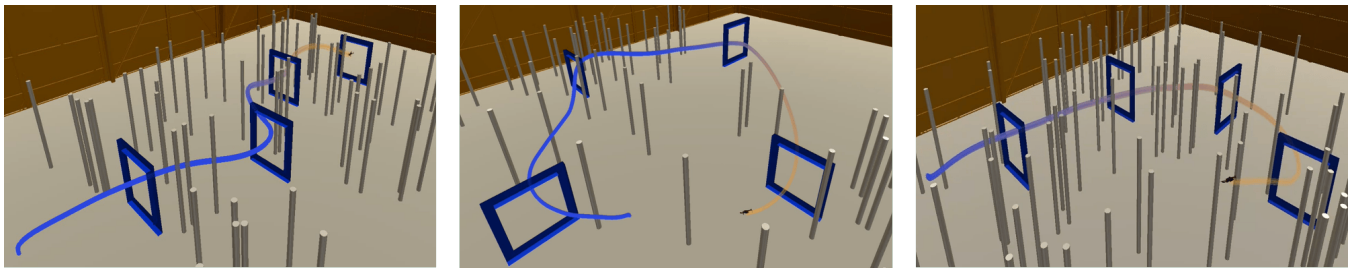

Fig. 4: **Three different racetracks with trajectories flown by our policy in simulation.** It can be observed that under our vision-based policy, the drone can successfully race in densely cluttered environments.

to address the conflict between gate passing and obstacle avoidance tasks. r_t^{pass} and r_t^{crash} are binary rewards that are active only when the drone successfully passes the next gate or a crash happens, respectively. The individual rewards are calculated as follows:

$$
\begin{aligned}
r_t^{\text{prog}} &= \lambda_1 \left(d_{t-1} - d_t\right), \\
r_t^{\text{theta}} &= \lambda_2 \cdot exp(-\left\|\theta_t - \theta_{\text{gate}}\right\|), \\
r_t^{\text{cmd}} &= \lambda_3 \left\|\boldsymbol{u}_t\right\| + \lambda_4 \left\|\boldsymbol{u}_t - \boldsymbol{u}_{t-1}\right\|, \\
r_t^{\text{vd}} &= \lambda_5(\left\|\boldsymbol{v}_t\right\| - v_d), \\
r_t^{\text{avoid}} &= \lambda_6 \left(\frac{1}{d_{col} + b_\omega}\right), \\
r_t^{\text{pass}} &= \begin{cases} \lambda_7 & \text{if drone passes the next gate} \\ 0 & \text{otherwise} \end{cases} \\
r_t^{\text{crash}} &= \begin{cases} \lambda_8 & \text{if collision occurs} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{5}
$$

where d_t represents the distance between the drone and the next gate centers. θ_t and θ_{gate} represent the yaw angle of the drone and the angle from the drone to the next gate's center in the x-y plane, respectively. $\boldsymbol{u}_t$ denotes the policy's output. $\boldsymbol{v}_t$ represent the linear velocity and v_d is the desired speed. d_{col} represents the distance between the drone and the nearest collision point, b_ω is a minor adjustment number.

4) Network Architecture: As illustrated in Fig. 2, the end-to-end architecture comprises a multi-modal feature extractor and an actor-critic network. Specifically, state-based observations and depth maps are encoded by a two-layer MLP and a three-layer CNN, respectively. The resulting feature vectors are concatenated and fed into a Gated Recurrent Unit (GRU) [21] to capture temporal dependencies. The actor subsequently generates high-level Collective Thrust and Body Rate (CTBR) commands, facilitating direct sim-to-real deployment.

C. Policy Training

We adopt the proximal policy optimization (PPO) algorithm [22] for policy training, with the key hyperparameters summarized in Tab. I. To achieve zero-shot transfer to the real world, the policy must thoroughly explore the observation space and maintain robustness against the sim-to-real gap. We address these requirements by integrating multi-stage curriculum learning with domain randomization techniques.

1) Multi-stage Curriculum Learning: Directly training with a one-step approach causes the RL policy to fall into local optima. Therefore, we propose to design a curriculum to allow the agent to start by completing simple tasks and eventually achieve racing with random obstacles. The designed curriculum has three levels of increasing difficulty to guide the agent's learning process:

- Level 1: We train the policy at a lower speed in each racetrack for obstacle-free scenes with vision-based input and whole reward until the agent can finish the racing task.
- Level 2: We randomly place obstacles among the waypoints in each racetrack and train the agents to fly at a low desired speed until they can finish the racetrack.
- Level 3: Finally, we improve the original desired speed and remove the velocity penalty r^{vd}. We randomize the obstacles within the racetrack and increase their density, training the agents until they have the capability to finish the racetrack at maximum velocity.

2) Domain Randomization: Similar to the RL agents presented in previous works [18], [20], we design a domain randomization strategy to robustify a control policy against the sim-to-real gap. First, our strategy needs to account for the stochastic variations of obstacles, enabling the policy to learn a wider range of collision avoidance scenarios. Second, we consider the randomization of gate positions and the initial points of the drone, which is crucial for achieving effective sim-to-real transfer. Therefore, we subdivide our domain randomization approach into two parts: obstacle generator and random initialization strategy.

Obstacle Generator We design an obstacle generator to generate random obstacles at each racetrack. Specifically, we regard the area between every two gates as a cuboid and generate a fixed number of obstacles within this space using a random distribution. Our generator ensures that obstacles are always placed on the sections of the racetrack, with a safety distance of $0.5m$ around each initial point, while guaranteeing the existence of a traversable path through the environment at all times. During training, we reset the environment after a fixed number of time steps to enable the agent to learn from randomized scenarios. Additionally, our generator can specify the number and three-dimensional shapes of each group of obstacles to control the difficulty of the training curriculum. In our simulation, obstacles are

modeled as simple geometric primitives with physical collision properties to maintain computational efficiency during training and facilitate reliable sim-to-real transfer.

Random Initialization Strategy To enhance training efficiency and mitigate the risk of the agent converging to local optima, we ensure that the onboard camera in the simulation captures diverse gate configurations from various initial viewpoints, with all gates within each racetrack being fully visible from these randomized starting points. Concurrently, the initial pose of the drone is uniformly sampled around each starting point, incorporating variations of $\pm 0.5m$ in the x and y axes, and $\pm 0.5m$ in the z axis. To further improve robustness against uncertainties in gate positioning, we introduced additional randomization to the gate locations, with deviations of $\pm 1m$ in the x and y axes, and $\pm 0.3m$ in the z axis.

3) Multi-scene Updating: As shown in Fig. 5, we introduce a multi-scene updating technique to significantly enhance RL training efficiency. Traditional RL-based approaches that rely on parallel single-scene updating, where all agents simultaneously update to the same training scene after each rollout [5], [6]. Our method employs a variable number of scenes to update, which supports adjusting the difficulty of curriculum training by customizing the number of parallel-updated scenes, and all agents are trained in groups according to a predefined number of scenes. This strategy not only improves the robustness of the training process but also accelerates convergence. It is worth mentioning that this method effectively balances the exploration in the early stages of RL with the exploitation in the later stages. It can effectively mitigate policy overfitting, enabling agents to better adapt to unknown environments and achieve superior performance in obstacle-rich racing tasks.

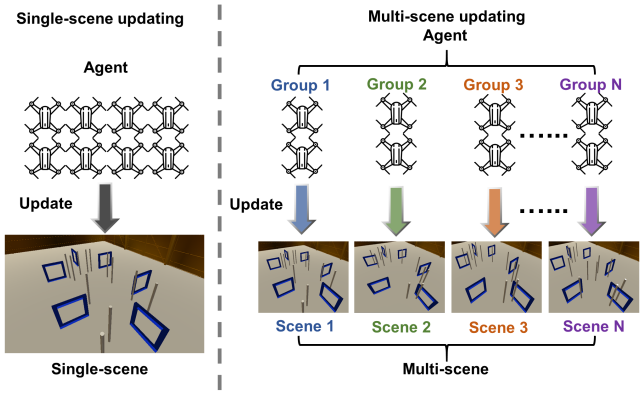

Fig. 5: **The framework of multi-scene updating.** The multi-scene updating approach improves policy training efficiency by customizing the number of scenes in each agent group.

IV. Experiments

A. Experimental Setup

We conduct experiments using the VisFly [23], a versatile quadrotor simulator for RL policy training with fast rendering based on Habitat-Sim [24]. We set up three different racetracks (S-shaped, J-shaped, 3D Circle) in obstacle-rich environments. The training is entirely done in simulation, where policy is trained for a total of 100 million time steps in each obstacle-rich racetrack. All the virtual gates and obstacles are equipped with a 3D physics engine and collision detection.

We assess our policy on each racetrack using two metrics: lap time (LT) and success rate (SR). LT measures the racing speed as the total time to complete one full lap, while SR reflects robustness by computing the proportion of successful laps without collision across 10 trials. All experiments run on an Ubuntu 20.04 machine equipped with an i9-13900K processor and an RTX-4090 GPU, achieving a rendering speed of approximately 6000 frames per second for visual inputs.

TABLE I: Key parameters during the experiment.

	Parameter	Value
Reward	λ_1	0.9
	λ_2	0.05
	λ_3	-0.005
	λ_4	-0.0025
	λ_5	-0.05
	λ_6	-0.01
	λ_7	5
	λ_8	-4
PPO	learning rate	1e-4 $\rightarrow$ 1e-5
	discount factor	0.99
	clip range	0.2
	GAE-λ	0.95
	batch size	51200
	number of parallel envs	100
	policy network MLP	[192,96]
	value network MLP	[192,96]
	GRU latent dimension	256
	number of update scenes	10
Quadrotor	mass [kg]	0.58
	inertia [g m^2]	[1.01, 1.53, 2.03]
	maximum thrust [N]	14
	arm length [m]	0.075

B. Baseline Comparison for Obstacle-aware Racing

To evaluate the performance of our policy, we compare it with two baselines: a vision-based policy obtained via RL [5] and a state-based policy trained by our method. The compared vision-based method is also an RL-training policy for obstacle-aware racing, which is only deployed in simulation. In contrast to our method, this approach is to simultaneously perform domain randomization on the racetracks and obstacles during training. Instead of introducing a dedicated reward for obstacle avoidance, it relies solely on discrete penalty terms to learn generalization through trial and error. The state-based policy is our policy training without depth input, including relative positions, velocity, orientation, angle velocity, desired velocity, and the same reward and training techniques. To ensure the fairness of the tests, we employ the same parameter settings of PPO, the same seed, and identical domain randomization methods.

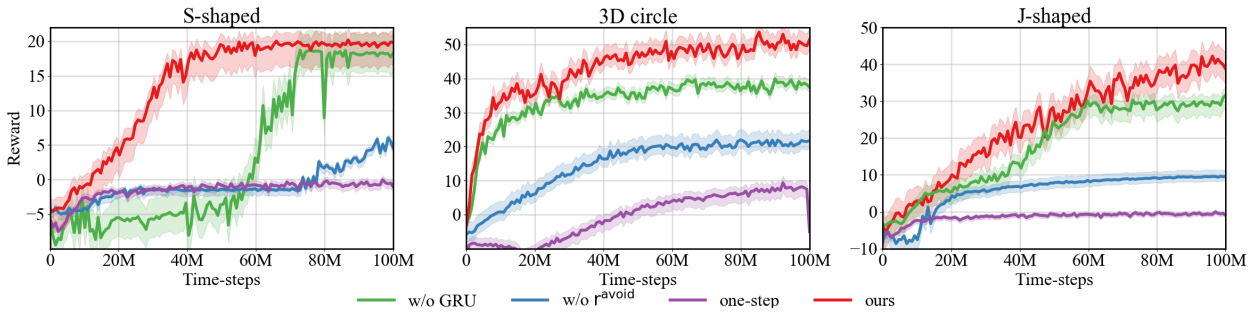

Fig. 6: **The reward curves of the ablation study during training.** Green, blue, purple, and red represent the cases without GRU, without r_t^{avoid}, one-step learning, and our method, respectively.

TABLE II: SR and the LT of our method against two baselines among 10 trials in simulation. Three different racetracks are used to evaluate the performance. The best results are bold.

Method	**LT [s]**			**SR [%]**		
	S-shaped	3D Circle	J-shaped	S-shaped	3D Circle	J-shaped
Vision-based [5]	5.4	3.8	3.9	30	40	40
State-based (ours)	3.5	3.8	3.1	30	20	30
Vision-based (ours)	**3.4**	**3.6**	**2.9**	**100**	**100**	**100**

The results are shown in Tab. II, our policy achieves 100% SR among three different racetracks, with 63% and 73% average higher than the other two methods. In terms of LT, our method is comparable to the state-based method, yet it outperforms the vision-based baseline [5], which is also designed for obstacle-aware racing. Especially in the curvilinear racetracks, our method achieves a significant performance. For the vision-based method [5], the lack of collision reward during training, combined with the absence of our multi-stage curriculum learning and multi-scene update strategy, results in poor generalization performance and a lower SR. The state-based method, although fast in speed, suffers from low SR due to the lack of visual input guidance for obstacle avoidance. Fig. 4 demonstrates the corresponding obstacle avoidance trajectories generated by our policy in highly cluttered racetracks.

C. Ablation Study

We conduct ablative studies to validate the design of our strategy, specifically focusing on three key components: the effectiveness of multi-stage curriculum learning in task completion, the impact of the obstacle-avoidance reward, and the role of GRU. These elements are designed to enable the agent to accomplish the complex task of racing in cluttered environments through progressively increasing difficulty, to learn obstacle avoidance in training, and to process visual-temporal input sequences, respectively.

Tab. III and Fig. 6 present the SR and reward curves of the ablation study, respectively. It can be observed that the performance of our policy deteriorates significantly when any one of the three components is ablated. Among these, replacing the multi-stage curriculum learning with a single one-step training leads to the most severe degradation, causing the policy to converge to a local optimum and fail to accomplish any tasks. Without r^{avoid}, the policy performance also declines notably, with the average SR across all three tracks dropping substantially to merely 36.7%. Similarly, when the GRU module is removed during training, the lack of temporal information results in slower convergence and reduced success rate. The ablation study validates the importance of the multi-stage curriculum learning, the obstacle avoidance reward, and the GRU module.

TABLE III: SR comparison for ablation studies among 10 trials.

		SR [%]		
		S-shaped	3D Circle	J-shaped
Study. 1	w/o GRU	80	70	80
Study. 2	w/o r^{avoid}	40	40	30
Study. 3	one-step	0	0	0
	ours	**100**	**100**	**100**

D. Handing RaceTracks and Obstacles Changes

Our objective is to address the task of racing with random obstacles. Therefore, it is essential to investigate the effectiveness of our method in variable obstacle environments and understand how the performance is influenced by the density of obstacles. Additionally, although this is not our primary focus, we also consider the variability of the racetracks by changing the positions of the gates. Specifically, we conduct two independent sets of evaluations: first, we increase the number of obstacles between every two gates across all tracks from 2 to 5. Second, we separately vary the spatial randomization level of the gates along each axis from $\pm 0.3\,m$ to $\pm 1.0\,m$.

The results are shown in Tab. IV. Our policy achieves SR of 76.7% and 70% on the most difficult test set,

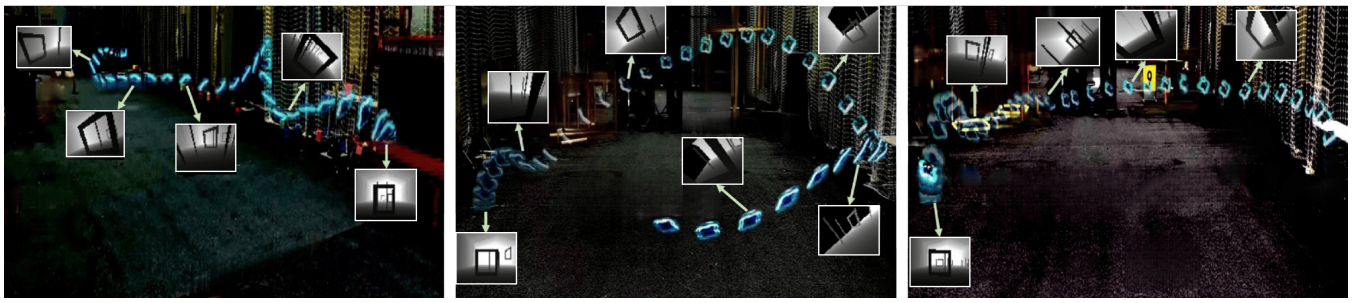

Fig. 7: **HITL simulation.** From left to right, S-shaped, 3D Circle, and J-shaped racetracks.

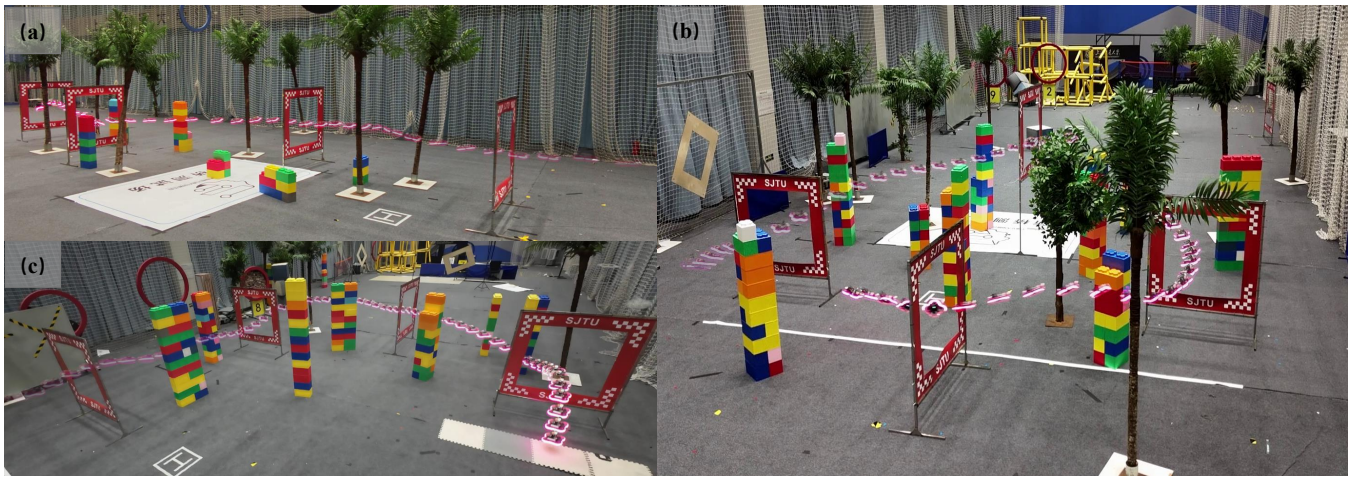

Fig. 8: **Real world experiment results.** We validate our algorithm in three racetracks, including (a) S-shaped, (b) 3D circle, (c) J-shaped.

which features maximum obstacle number and gates position changing. The results fully demonstrate that our strategy can robustly handle uncertain obstacle changes, exhibiting good performance even under conditions of dense obstacles and small racetrack adjustments.

TABLE IV: SR of handling gates and obstacles across 10 trials. The results correspond to independent evaluations of two factors: (1) Density, which refers to the number of obstacles affecting flight in the racetrack in every 2 gates; and (2) Gate changing (m), which denotes the gate's randomization level among the three axes.

		SR [%]		
		S-shaped	3D Circle	J-shaped
Density [its/2 gates]	2	100	100	100
	3	100	100	100
	4	90	90	90
	5	80	70	80
Gate changing [m]	± 0.3	100	100	100
	± 0.5	100	100	100
	± 0.7	90	80	90
	± 1.0	80	60	70

E. Hardware-in-the-loop Simulation

Hardware-in-the-loop (HITL) simulation refers to a method of validating policy using real-world dynamics and visual inputs in simulation, which has been proven to be effective [7]. To prevent damage to the onboard computing devices during the initial validation phase, we first employ the dynamics of the real drone and the depth maps corresponding to the positions in the VisFly, ensuring the safety of high-speed flight. Fig. 7 presents the HITL simulation results along with the first-person-view depth images. It shows that our policy is capable of racing through the gates of each racetrack with visual obstacles at a maximum desired speed v_d exceeding $10m/s$ while avoiding obstacles along the way. The SR and LT of the HITL simulation are shown in Tab. V.

TABLE V: The performance of our policy in HITL simulation and the real world.

Tracks	**S-shaped**		**3D Circle**		**J-shaped**	
	SR [%]	LT [s]	SR [%]	LT [s]	SR [%]	LT [s]
HITL	100	3.5	100	4.1	100	3.2
Real-world	100	4.3	100	5.0	100	3.7

F. Real-World Experiment

Finally, we validate our policy in the real world (see Fig. 1 for visualization). The entire control pipeline is executed via onboard inference on a Raspberry Pi. Depth data from an Intel D435i camera is downsampled and provided to the policy network at a frequency of 30 Hz. To ensure low-latency performance, the weights are exported in the ONNX

[25] format, enabling the network to generate control commands at a consistent rate of 30 Hz. These CTBR commands are then processed by the BetaFlight firmware for low-level control. All experiments are conducted in an indoor area using a Vicon motion capture system for state estimation. Each racetrack is evaluated through three separate trials under varying obstacle placements and initial conditions.

The real-world flight performance is shown in Tab. V and Fig. 8. For a more dynamic impression, we advise readers to watch the supplementary video showcasing these experiments. Our approach achieves zero-shot sim-to-real transfer of the end-to-end vision-based control policy by aligning real-world dynamics with those observed in simulation. Across three racetracks, the success rate reaches 100%. Taking into account the obstacle density and the hardware safety constraints of the physical platform, v_d is set to 8 m/s.

V. Limitations

In this work, we employ a curriculum RL method for obstacle-aware drone racing, which provides an effective reference for end-to-end control. Nevertheless, due to the inherent exploration and sample-efficiency limitations of RL, our policy can adapt to randomly placed obstacles but fails to generalize to unseen racetrack layouts. This limitation highlights an open problem for current RL-based approaches. In the future, combining the exploratory capacity of RL with the efficiency of differentiable physics learning may enable policies to generalize to random racetracks with obstacles.

VI. Conclusions

In this paper, we presented a vision-based curriculum reinforcement learning framework for drone racing in cluttered environments. By integrating a multi-stage curriculum with a multi-scene updating strategy, our approach enables agile flight that balances high-speed gate traversal and reactive obstacle avoidance. Our results demonstrate that the learned policy exhibits significant generalization capabilities across various random obstacle placements. While the current framework assumes a fixed track layout, it effectively accommodates minor perturbations in gate positioning, showing strong robustness during sim-to-real transfer. Future work will focus on extending this generalization to encompass entirely unseen racetrack layouts and more diverse gate geometries, further pushing the limits of autonomous flight in fully unstructured environments.

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*, vol. 40, pp. 3044–3067, 2024.

[2] 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.

[3] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[4] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for se(3) planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.

[5] Y. Liu, "Learning generalizable policy for obstacle-aware autonomous drone racing," 2024. [Online]. Available: https://arxiv.org/abs/2411.04246

[6] 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.

[7] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1989–1995.

[8] 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, Oct. 2021.

[9] H. Yu, C. D. Wagter, and G. C. H. E. de Croon, "MAVRL: learn to fly in cluttered environments with varying speed," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1441–1448, 2025.

[10] G. Zhao, T. Wu, Y. Chen, and F. Gao, "Learning speed adaptation for flight in clutter," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7222–7229, 2024.

[11] Y. Zhang, Y. Hu, Y. Song, Z. Danping, and W. Lin, "Learning vision-based agile flight via differentiable physics," *Nature Machine Intelligence*, pp. 1–13, 06 2025.

[12] Y. Hu, Y. Zhang, Y. Song, Y. Deng, F. Yu, L. Zhang, W. Lin, D. Zou, and W. Yu, "Seeing through pixel motion: Learning obstacle avoidance from optical flow with one camera," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5871–5878, 2025.

[13] 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, 2023.

[14] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[15] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1458–1465, 2018.

[16] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8.

[17] 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.

[18] 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.

[19] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," 2024. [Online]. Available: https://arxiv.org/abs/2406.12505

[20] J. Xing, I. Geles, Y. Song, E. Aljalbout, and D. Scaramuzza, "Multi-task reinforcement learning for quadrotors," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2112–2119, 2025.

[21] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014. [Online]. Available: https://arxiv.org/abs/1406.1078

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[23] F. Li, F. Sun, T. Zhang, and D. Zou, "Visfly: An efficient and versatile simulator for training vision-based flight," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 11 325–11 332.

[24] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9338–9346.

[25] ONNX Contributors, "Open neural network exchange (onnx)," https://github.com/onnx/onnx, 2021, version 1.10.0. Accessed: Sep. 15, 2025.