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

Agile Flight Emerges from Multi-Agent Competitive Racing

Vineet Pasumarti* Lorenzo Bianchi* Antonio Loquercio

***Abstract*— Through multi-agent competition and the sparse high-level objective of winning a race, we find that both agile flight (e.g., high-speed motion pushing the platform to its physical limits) and strategy (e.g., overtaking or blocking) emerge from agents trained with reinforcement learning. We provide evidence in both simulation and the real world that this approach outperforms the common paradigm of training agents in isolation with rewards that prescribe behavior, e.g., progress on the raceline, in particular when the complexity of the environment increases, e.g., in the presence of obstacles. Moreover, we find that multi-agent competition yields policies that transfer more reliably to the real world than policies trained with a single-agent progress-based reward, despite the two methods using the same simulation environment, randomization strategy, and hardware. In addition to improved sim-to-real transfer, the multi-agent policies also exhibit some degree of generalization to opponents unseen at training time. Overall, our work, following in the tradition of multi-agent competitive game-play in digital domains, shows that sparse task-level rewards are sufficient for training agents capable of advanced low-level control in the physical world.** Code Video

I. INTRODUCTION

Drone racing, a competitive sport where pilots fly quadcopters through tortuous circuits at high speed, has become a widely adopted benchmark for autonomous control [1]. Indeed, its requirement for decision-making under tight time constraints with little tolerance for errors makes it an ideal setting for testing advanced control strategies. In recent years, reinforcement learning (RL) has shown remarkable success on this task, consistently outperforming classical optimal-control techniques and even rivaling champion-level human pilots [2]–[4]. RL's strength lies in its ability to optimize over long horizons and high-dimensional inputs. Such capabilities are difficult to achieve with model-based controllers, given their heavy reliance on online optimization.

Despite this promise, most RL approaches for drone racing remain more closely tied to optimal control than is often acknowledged. In particular, they typically require dense shaping rewards such as progress on the segment connecting two consecutive gates [2]–[6], which closely resembles trajectory-tracking costs used in optimal controllers [7]–[9]. As a result, most prior work trains RL agents to effectively follow a race line as fast as possible. This strategy prioritizes speed but prescribes behavior in a way that is not fundamentally different from trajectory optimization. Yet, as insights from game theory suggest [10]–[12], maximizing speed alone is not always the optimal strategy for winning a race. Successful racing also requires tactical behaviors such as overtaking, blocking, or avoiding collisions, which are not easily captured by progress-based costs.

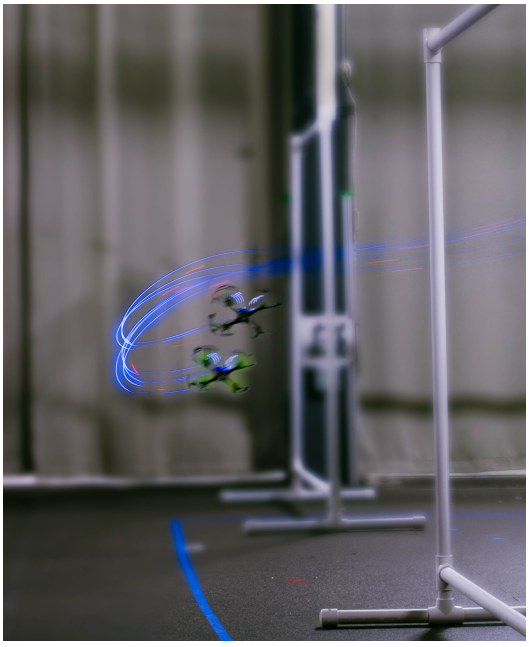

Fig. 1: Two opponent-aware quadrotors performing head-to-head autonomous racing. The multi-agent policies are trained in simulation with competitive, sparse, task-level rewards (winning the race), without behavior-based rewards (flying fast), and are transferred zero-shot to real-world drones.

This observation raises the question: *can RL agents learn racing strategies directly from outcome-based objectives, such as winning, without relying on dense behavioral shaping?* Our key insight is that this is possible when drone racing is framed as a multi-agent problem. By rewarding agents only for finishing a lap before their opponent, we show that fast and *agile flight emerges naturally from sparse, competition-based rewards*. Moreover, this formulation gives rise to sophisticated behaviors, *e.g.*, overtaking, blocking, and collision avoidance, despite lacking explicit reward terms for these skills. In contrast, we find that dense progress-based rewards, even when supplemented with overtaking terms [6], [13], are limiting as the complexity of the track increases, *e.g.*, in the presence of obstacles, because their prescriptive structure constrains exploration during training.

Our work builds on the tradition of learning from multi-agent interaction [14]–[18], where rich behaviors emerge from simple task-level competitive rewards. However, in contrast to prior studies that primarily examined abstract or simplified environments, we observe a similar effect, albeit at a smaller scale, in a physically realistic, embodied setting,

* denotes equal contribution. LB is affiliated to the University of Rome Tor Vergata. VP and AL are with the University of Pennsylvania.

where learned policies can be directly deployed to real-world drones. Interestingly, we find that policies trained with sparse multi-agent rewards transfer more reliably to the real world than the ones trained with dense progress rewards, despite the two methods sharing the same simulation environment, randomization strategy, policy structure, and physical hardware. With our work, we aim to inspire a shift in perspective within the real-world control community: from designing controllers that prescribe specific behaviors to designing controllers that optimize task-level objectives, letting desired behaviors emerge naturally.

Contributions. (1) We show that formulating autonomous drone racing as a competitive multi-agent problem naturally induces agile flight and tactical behaviors without explicit behavioral shaping. (2) We demonstrate that this approach outperforms dense progress-based rewards, in particular as the track complexity increases, while transferring better to the real world, and (3) We show generalization of our policies to agents it was not trained on.

II. RELATED WORK

A. Single-Agent Drone Racing

A wide range of controllers has been explored to tackle drone racing, ranging from classical methods to approaches based on deep learning [1].

Among classical approaches, Model Predictive Control (MPC) and its variants are the most widely adopted. In [9], an MPC controller tracks a time-optimal trajectory computed offline. However, disturbances may cause suboptimal performance. To reduce this sensitivity, Model Predictive Contouring Control (MPCC) [7], [8] performs online adaptation of path, velocities, and accelerations based on a track reference computed offline, leading to greater robustness.

While effective, the previous approaches face two main challenges: they require an accurate drone model, which is difficult to obtain, especially when considering aerodynamic effects during complex maneuvers; also, they often require significant computation power at inference time, which requires them to optimize relatively short-horizon objectives. A natural solution to these challenges is provided by RL-based controllers [2], [4], [5], [19], [20]. This approach waives the requirement of a very accurate model [21] and removes strict separation between long-term planning and control.

However, these methods focus on single-drone performance and do not study interaction dynamics between multiple agents. Additionally, they require carefully designed reward functions to perform well. In this work, we show that these two issues are related: considering adversarial interaction directly leads to simplified reward shaping.

B. Multi-Agent Drone Racing

The literature on multi-drone autonomous racing is much sparser than that on single-agent racing. Existing approaches rely on different strategies. To model the interaction and compute the best response, some methods rely on MPC [22] or game-theoretic formulations [10], [23], [24]. However, these methods face severe scalability issues, as online computational cost increases with the complexity of the environment, which limits them to relatively low-speed racing.

Reinforcement learning offers an alternative way to address these limitations. Prior works trained multi-agent policies by augmenting single-agent rewards with an additional 'overtaking' term [6], [25], following approaches used in multi-agent car racing for Gran Turismo [13]. In contrast, we show that dense single-agent rewards are not only unnecessary—since agile flight naturally emerges from the competitive dynamics of racing—but can even degrade performance as track complexity increases (e.g., with obstacles).

Conceptually, our work is closely related to research on multi-agent games, such as Capture the Flag [15], StarCraft [26], hide-and-seek [27], and autonomous driving [18]. Similar to our approach, these studies demonstrate that training reinforcement learning agents with simple adversarial rewards at scale can lead to complex emergent behaviors, such as the use of tools [27] or the development of social norms [18]. However, these works primarily focus on high-level strategy and abstract away low-level dynamics, which confines them to simulation environments. In contrast, our work shows that interesting behaviors also emerge when training policies on realistic dynamical systems, enabling zero-shot transfer of the policies to real hardware.

III. METHODOLOGY

In the following, we detail the policy optimization procedure and describe the dynamics model used for training.

A. Multi-Agent Policy Optimization

We define drone racing as a multi-agent general-sum game between two agents π^{e} and π^{a}, referred in the following as the *ego* and *adversary*. We jointly optimize the policies to maximize the discrete-time finite-horizon objective:

$$J(\pi^e, \pi^a) = \mathbb{E}_{\tau \sim p(\tau \mid \pi^e, \pi^a)} \Big[\sum\nolimits_{t=0}^{T} \gamma^t r_t(x_t^e, x_t^a) \Big], \quad (1)$$

where T is the finite time horizon, and r_t is a competitive sparse task reward. Here, the joint trajectory of states, actions, and rewards $\tau = \{(\boldsymbol{x}_t^{\mathrm{e}}, \boldsymbol{x}_t^{\mathrm{a}}, \boldsymbol{u}_t^{\mathrm{e}}, \boldsymbol{u}_t^{\mathrm{a}}, r_t)\}_{t=0}^{T}$ is induced by the ego and adversary policies, drawn from the distribution $p(\tau \mid \pi^{\mathrm{e}}, \pi^{\mathrm{a}})$. The discount factor is γ. Formally, this optimization defines the solution to a two-agent, finite-horizon decentralized Markov decision process (Dec-MDP).

We denote the agents' global physical state as $\boldsymbol{x}^{\mathrm{e}}, \boldsymbol{x}^{\mathrm{a}} \in \mathbb{R}^n$ (with n varying according to the environment). Similarly to prior works [2], [3], the policies are reactive and predict the drones' vertical thrust and body rates, i.e., $\boldsymbol{u}^{\mathrm{a}}, \boldsymbol{u}^{\mathrm{e}} \in \mathcal{U} = [f_z^{des}, \boldsymbol{\omega}_{des}]$ (see Sec. III-B). A fixed low-level PID controller then maps these to rotor forces (see Sec. III-C)

Rewards. We use a sparse reward structure encouraging emergent racing behaviors without explicitly defining strategies: the agents are rewarded for passing a gate before their opponent (r_t^{pass}) with a bonus for completing a full lap first (r_t^{lap}). We also add an energy minimization term (r_t^{cmd}) for

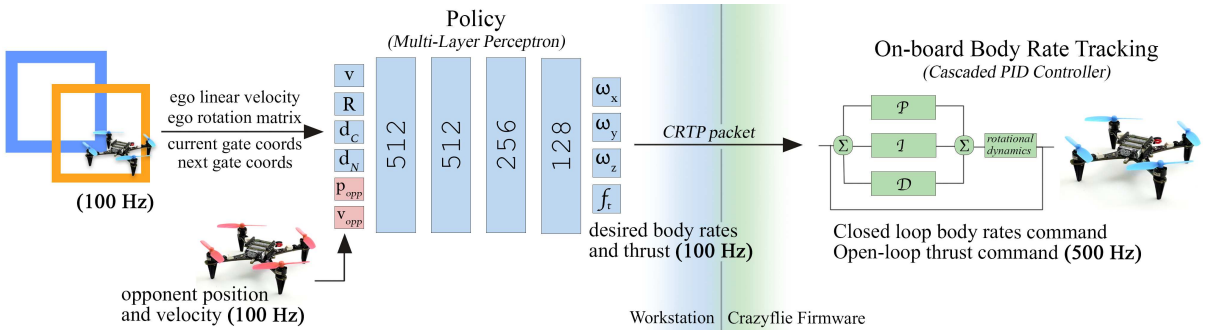

Fig. 2: Observation to motor-command pipeline. Each drone's actor network receives ego-centric and opponent state estimates at 100 Hz from the Vicon motion capture system. A multi-layer perceptron processes the observations and outputs desired body rates and thrust, which are then sent to the drone via Crazy Real-Time Protocol (CRTP).

regularization, as well as a penalty for crashing into the ground or out of bounds (r_t^{crash}). Therefore,

$$r_t = r_t^{\text{pass}} + r_t^{\text{lap}} - r_t^{\text{cmd}} - r_t^{\text{crash}}, \tag{2}$$

$$r_t^{\text{pass}} = \begin{cases} 10.0, & \text{if gate passed at } t \text{ and leading} \\ 5.0, & \text{elif passed gate at timestep} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$r_t^{\text{lap}} = \begin{cases} 50.0, & \text{if lap completed at } t \text{ and leading} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$r_t^{\text{cmd}} = -0.15\,(\omega_{roll}^2 + \omega_{pitch}^2) - 0.05\,\omega_{yaw}^2 \tag{5}$$

$$r_t^{\text{crash}} = \begin{cases} 2.0, & \text{if terminally crashed} \\ 0.1, & \text{elif in contact} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where ω_{roll}, ω_{pitch}, and ω_{yaw} are the commanded body rates. The reward scales are chosen empirically and we do not modify the environment in any way by adding extra "phantom" gates to bias the drone trajectory.

Comparison to Existing Rewards. Our reward design removes a key component used in existing approaches: *race-line progress*. This is typically defined as

$$r_t^{\text{Prog}} = d_{t-1}^{\text{Gate}} - d_t^{\text{Gate}}, \tag{7}$$

where d_t^{Gate} denotes the distance to the center of the next gate at time t. Intuitively, this dense reward encourages the agent to reach the next gate as quickly as possible while remaining close to the straight line connecting consecutive gates. However, we argue that such a formulation, reminiscent of reference trajectories in model-based controllers [7], is overly restrictive. For example, it discourages deviations needed for obstacle avoidance and penalizes higher-level strategies such as blocking, overtaking, or adapting to an opponent's crash. While existing methods have tried to overcome these issues in multi-agent settings by adding overtaking rewards [6], [13], [25], we found this to be inferior to our approach, especially in the presence of obstacles.

Optimization We jointly train π^{e} and π^{a} to optimize Eq. 1 using IPPO [28], a multi-agent variant of PPO [29]. IPPO differs from standard PPO in that the surrogate objective is computed as the average of the individual agents' surrogate losses. Unlike MAPPO, IPPO does not employ a shared critic, which would require larger networks and potentially longer trainings. Instead, each agent maintains its own policy and critic, a design that we found to yield strong empirical performance. For this reason, and to keep the method simple, we did not explore alternative multi-agent RL formulations.

B. Actor and Critic Parametrization

Actor Network. Both agents are parametrized as MLPs with the same architecture and observation format. We use a 42-dimensional state vector defined as $\mathbf{x}_t^a, \mathbf{x}_t^e = [\mathbf{v}, \mathbf{R}, \mathbf{d}_C, \mathbf{d}_N, \mathbf{p}_{\text{opp}}, \mathbf{v}_{\text{opp}}]$, where $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity of the drone expressed in body frame, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is the attitude rotation matrix, $\mathbf{d}_C, \mathbf{d}_N \in \mathbb{R}^{12}$ are the position of the current and next gates' corners in body frame, respectively, and $\mathbf{p}_{\text{opp}}, \mathbf{v}_{\text{opp}} \in \mathbb{R}^3$ are the opponent position and linear velocity in body frame, which are provided by a Vicon motion capture system. In the tracks with obstacles, we extend the state space to $\mathbf{x}_t^a, \mathbf{x}_t^e, \in \mathbb{R}^{45}$ by including the global position $\mathbf{p} \in \mathbb{R}^3$ of the drone. Both policy networks have dimensions [512, 512, 256, 128] with ELU activation.

Critic Network For simplicity, we use separate critic networks for the two agents. Each critic receives privileged input in the form of the concatenated joint state, $x_{\text{critic}} = [\boldsymbol{x}^{\text{e}}, \boldsymbol{x}^{\text{a}}]$, and is implemented as a fully connected MLP. The hidden layers have dimensions [512, 512, 256, 256, 128, 128] with ELU activations. We empirically find that making the critic deeper than the actor improves performance.

C. Quadrotor Simulation

We train π^{e} and π^{a} exclusively in simulation and deploy them zero-shot to the real world. To keep the approach simple, we avoid extensive system identification and instead follow prior work [4], [21] by relying on domain randomization during training and rapid adaptation at test time. Our quadrotor dynamics model is based on the one in [30], with an additional aerodynamic component to capture drag, implemented following the model proposed in [31].

Our quadrotor simulation in Isaac Sim [32] models a cascaded control architecture. A high-level controller, running at 50 Hz, generates thrust and angular rate commands, while a low-level controller tracks the desired body rates using a PID at 500 Hz. These commands are converted into motor

forces and torques and applied, together with aerodynamic effects, to the quadcopter, modeled as a rigid body.

Denote $[a_0, a_1, a_2, a_3]$ a policy output. The first element $a_0 \in [-1, 1]$ is mapped to the thrust along the body z-axis:

$$f_z^{\text{des}} = (a_0 + 1) \,/\, 2 \cdot T, \tag{8}$$

where T is the maximum thrust expressed in units of drone weight. The remaining action components $a_1, a_2, a_3 \in [-1, 1]$ determine the desired angular velocities:

$$\boldsymbol{\omega}^{\text{des}} = \left[k_x a_1,\, k_y a_2,\, k_z a_3\right], \tag{9}$$

where k_x, k_y, and k_z are the maximum roll, pitch, and yaw rate, respectively. The angular velocity error $\boldsymbol{e}_\omega = \boldsymbol{\omega}^{\text{des}} - \boldsymbol{\omega}$ is then converted to desired torques via a PID control law, after scaling by the inertia matrix $\mathcal{I}$:

$$\boldsymbol{\tau}^{\text{des}} = \mathcal{I}\left(K_p \boldsymbol{e}_\omega - K_d \dot{\boldsymbol{e}}_\omega + K_I \textstyle\int \boldsymbol{e}_\omega\right). \tag{10}$$

The desired wrench $\boldsymbol{w}^{\text{des}} = [f_z^{\text{des}}, \boldsymbol{\tau}^{\text{des}}]$ applied to the rigid body is then converted to motor forces by preconditioning with the inverse of the thrust-to-wrench static mapping, i.e., $\boldsymbol{f}_{\text{mot}}^{\text{des}} = T_M^{-T} \cdot \boldsymbol{w}^{\text{des}}$. From these forces, the desired motor speeds for each motor $i = 1, \ldots, 4$ are computed using the thrust coefficient k_η, i.e., $\boldsymbol{\omega}_{\text{mot}}^2 = \boldsymbol{f}_{\text{mot}}^{\text{des}} \,/\, k_\eta$. These are then converted to actual motor speeds using the minimum and maximum allowable motor speeds, $\underline{\omega}$ and $\bar{\omega}$:

$$\omega_{\text{mot, i}}^{\text{des}} = \omega_{\text{i}}^{\text{des}} = \text{clamp}\left(\sqrt{|\omega_i^2|} \cdot \text{sign}(\omega_i), \underline{\omega}, \bar{\omega}\right). \tag{11}$$

We model the motor dynamics with a first-order model governed by the motor constant τ_m:

$$\dot{\omega}_i = \left(\omega_i^{\text{des}} - \omega_i\right) \,/\, \tau_m, \quad \omega_i \leftarrow \omega_i + \dot{\omega}_i \cdot \Delta t. \tag{12}$$

Once we compute the actual motor speed, we compute the actual forces and wrench applied to the body as

$$f_i = k_\eta \cdot \omega_i^2, \quad \boldsymbol{w} = T_M \cdot \boldsymbol{f} = \left[f_z, \boldsymbol{\tau}^T\right]^T. \tag{13}$$

In addition, similarly to [31], we model aerodynamic effects as forces and torques proportional to the translational and angular velocities:

$$\boldsymbol{f}^{\text{drag}} = -\sum \omega_i \, \boldsymbol{K}_{\text{aero}} \, \boldsymbol{v}_b \tag{14}$$

where $\boldsymbol{K}_{\text{aero}}$ is a diagonal matrix and $\boldsymbol{v}_b$ is the linear velocity of the drone expressed in the body frame. The final applied force and torque acting on the rigid body are:

$$\boldsymbol{F}_{\text{app}} = \left[f_x^{\text{drag}}, f_y^{\text{drag}}, f_z^{\text{drag}} + f_z\right]^T, \quad \boldsymbol{\tau}_{\text{app}} = \boldsymbol{\tau}. \tag{15}$$

IV. Experiments

Our evaluation is designed to find answers to the following questions. What are the limitations of single-agent racing rewards? How do agents perform in head-to-head races? Can multi-agent policies bridge the gap between controlled experiments and deployment on physical platforms? Does a qualitative analysis reveal the emergence of strategic behaviors? Finally, we study the stability of training multi-agent policies with sparse rewards.

A. Experimental Setup

We simulate a custom Crazyflie 2.1 Brushless environment in Isaac Lab v2.2.0 and Isaac Sim v4.5.0 [32]. This quadrotor was chosen for its compact form factor ($3''$), lightweight design (32 g), and high thrust-to-weight ratio (slightly greater than 3). We also designed two tracks: the *Complex Track* (CT) and the *Lemniscate Track* (LT), shown in Fig. 3, along with the gate-passing directions, obstacles, and a drone trajectory. The CT consists of six gates (including a split-S maneuver) and four obstacles, while the LT features five gates (one crossed twice) and two obstacles. Both track layouts were designed to be challenging enough to require agile motions, even in the absence of obstacles. Additionally, when obstacles are present, their placement forces the drones to first move away from the next gate before reaching it. In the real-world setup, flights were carried out within a controlled indoor area, measuring $22.0 \text{ m} \times 5.5 \text{ m} \times 3.8 \text{ m}$. A motion capture system with 25 Vicon cameras provided ground-truth poses at a frequency above 100 Hz.

We benchmark against four policies varying by training setup (Single-/Multi-Agent) and reward (dense/sparse): *Dense Single-Agent* (DS), similar to [2]–[6]; *Sparse Single-Agent* (SS), using our same sparse reward but trained without a competitor; *Dense Multi-Agent* (DM), which, similarly to [6], [13], complements the progress reward with the dense overtaking term, with $d_{min} = -5$ m and $d_{max} = 10$ m,

$$r_t^{\text{ot}} = \mathbb{I}_{d_{min} < (p_t^i - p_t) < d_{max}} \left((p_t - p_t^i) - (p_{t-1} - p_{t-1}^i)\right);$$

and *Sparse Multi-Agent* (Ours). In the multi-agent cases, we evaluate the actor that received the greatest total reward during training and discard the other. All baselines are carefully tuned to achieve optimal performance.

To assess policy effectiveness, we performed two types of tests. First, we evaluated the single-agent policies in simulation across various track layouts and obstacle configurations. Agents were not directly provided with the obstacle positions; instead, the algorithm inferred them by exploiting the global drone positions. As a result, eight independent experiments were conducted, each consisting of fifty separate 3-lap runs with different initial positions beyond a selected gate. Four metrics were used for evaluation: the average Time To Complete (TTC) one lap in seconds, the average speed in meters per second (both computed over the fifty runs), the total number of collisions per experiment against gates and obstacles, and the success rate, measured as the proportion of 3-lap runs successfully completed out of fifty trials.

The second set of tests focused on head-to-head races between all pairs of trained policies across all the track layouts. Additionally, several multi-agent experiments were conducted in the real world using the lemniscate track. In both cases, the primary metric was the win rate of one policy against the other, expressed as a percentage of the total number of races, where each race consists of 3 laps. We do fifty races in simulation and three in the real world. The first agent to complete three laps is considered the winner, while a terminal crash by both agents is counted as a draw.

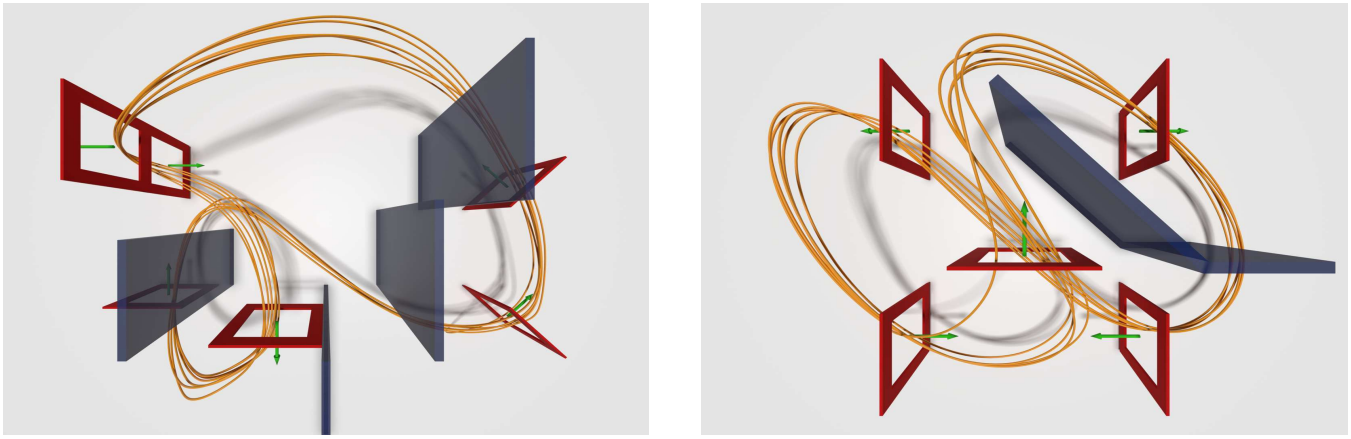

Fig. 3: Tracks used for training and evaluation, with $1\ \mathrm{m} \times 1\ \mathrm{m}$ gates in red and obstacles in transparent blue. Green arrows indicate the gate-passing directions, and the orange curves show the trajectories over multiple laps. On the left, the Complex Track (CT) spans $8\ \mathrm{m} \times 7\ \mathrm{m}$ with six gates, including a split-S, and optionally four obstacles. On the right, the Lemniscate Track (LT) measures $5\ \mathrm{m} \times 5\ \mathrm{m}$, with five gates, one of which passed twice in a single lap, and optionally two obstacles.

B. What are the limits of single-agent racing?

Table I presents the single-agent policy results. Notably, the policies trained with dense rewards achieved strong performance only in environments without obstacles, with a success rate of 100% on the lemniscate track and 98% on the complex track. In contrast, performance with obstacles is extremely poor, yielding a success rate of zero on both tracks, even failing to complete a single lap. This behavior is inherently due to Eq. 7, the progress reward, which discourages the drone from moving away from the gate, preventing it from successfully overcoming obstacles.

As for the sparse reward policies, their success rate on obstacle-free tracks is at times comparable to that of the dense policies, although with lower speed. In the lemniscate track, the average lap time of the SS policy is 5.29 s, compared to 4.76 s for DS, corresponding to an increase of 11.13%, while the average speed drops from 4.07 $\mathrm{m\,s^{-1}}$ to 3.57 $\mathrm{m\,s^{-1}}$ (-12.3%). In the complex track, the results are similar, with the lap time increasing from 5.68 s to 6.42 s (+13.0%) and the speed decreasing from 5.01 to 4.30 $\mathrm{m\,s^{-1}}$ (-14.17%). Additionally, applying the sparse reward on the lemniscate track with obstacles provides a success rate of 98%, whereas, on the complex track, it drops to 0%, despite the completion of a few individual laps.

Overall, these experiments show that in single-agent drone racing environments, dense rewards more reliably converge to competitive policies than sparse rewards, confirming the finding of prior work [2], [3]. Neither sparse nor dense rewards, however, consistently discover obstacle-free trajectories across tracks. Additionally, dense rewards are challenging to tune as the track complexity increases, e.g., in the presence of obstacles. While for static obstacles (as in our experiments) one could circumvent this problem by first doing coarse path planning and tracking the planned path, this solution is not general. For example, this heuristic would fail if the track requires avoidance of dynamic obstacles.

	Avg. TTC [s]	Avg. Speed [$\mathrm{m\,s^{-1}}$]	Collisions	Success [%]	
LT	4.76	4.07	0	100	Dense
LT w/ obs	–	–	175	0	
CT	5.68	5.01	1	98	
CT w/ obs	–	–	0	0	
LT	5.29	3.57	0	100	Sparse
LT w/ obs	5.72	3.95	12	98	
CT	6.42	4.30	16	94	
CT w/ obs	10.08	4.58	69	0	

TABLE I: Single-agent simulation results indicate a clear preference for dense rewards in obstacle-free environments.

C. Baseline comparison in simulated head-to-head races

We evaluated our multi-agent policy's competitive performance through head-to-head races, as it provides a clearer indication of dominance than lap times alone. We pitted two drones against each other in every race, running one of the four methods, and used the same single-agent policies that were evaluated in the previous section.

The results of this evaluation are shown in Fig. 4, where each cell in the four matrices reports the win rate of the row policy against the column policy. The entries are symmetric with respect to the main diagonal, corresponding to the same policy pair with reversed roles. Note that these percentages do not sum to 100%, as the remainder corresponds to the fraction of draws due to collision. We omit elements on the diagonal, as we don't evaluate a policy against itself.

These experiments confirm that in single-agent environments, simple sparse rewards rarely outperform dense rewards. The only scenario in which the SS policy shows a slight advantage is in the presence of obstacles, where it occasionally wins against DS. Extending the DS formulation to a multi-agent approach (DM) by adding a competition reward does not improve performance; in fact, it often results in more defeats compared to DS and still fails in the presence of obstacles. This outcome likely stems from the difficulty of balancing two dense reward terms, which could cause the policy to converge to a suboptimal local minimum.

Conversely, training with sparse competitive rewards (Ours) yields the highest average win rate of 91.17%. Interestingly, we outperform the dense baselines even on obstacle-free tracks, where we win 100% (50/50) of races on the

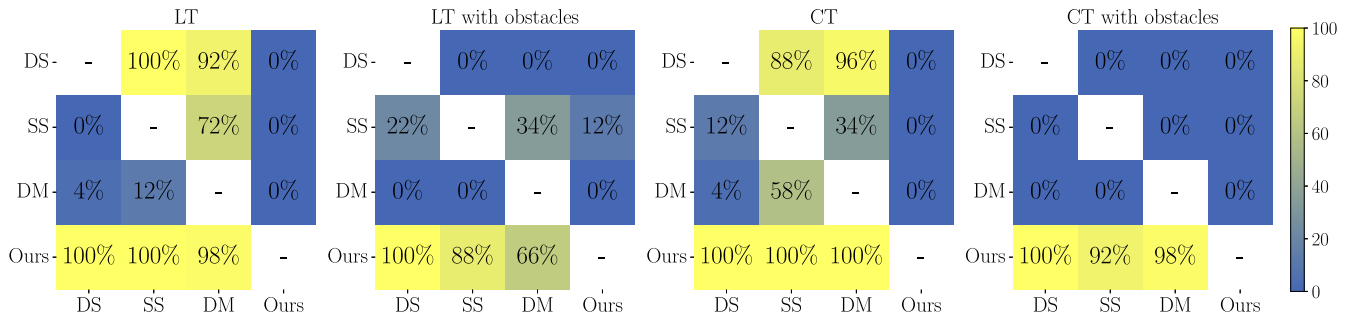

Fig. 4: Head-to-head results in simulation by varying track layouts and racing policies. The latter are defined as Dense Single-agent (DS) [2]–[6], Sparse Single-Agent (SS), an ablation of our approach without competition, Dense Multi-Agent (DM), similar to DS with a competitive reward, and Sparse Multi-Agent (Ours). Each cell shows the win rate of the row policy against the column policy.

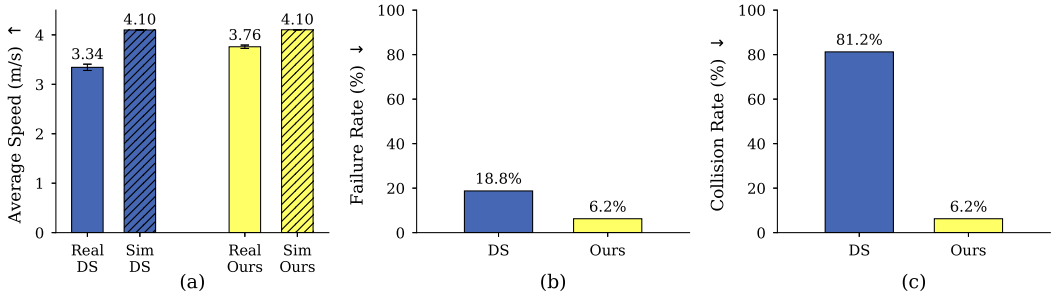

Fig. 5: Sim-to-real transfer evaluation (DS vs. Ours) using identical simulation and hardware. (a) Average in-flight speed comparison between simulation and real-world deployment on LT. (b) Failure rate (*terminal collisions / attempted laps*). (c) Collision rate (*collisions / attempted laps*). While simulated failure and collision rates are zero for both, Ours exhibits superior sim-to-real transfer, evidenced by a smaller sim-to-real speed gap and lower real-world failure and collision rates.

lemniscate and 84% (42/50) on the complex against DS. Occasionally, the win rate drops (62% and 66% against SS and DM, respectively, on the lemniscate), often due to one of the agents colliding or behaving in an erratic way that our policy did not observe at training time. The success of Ours on obstacle-laden tracks shows how competitive pressure among agents broadens the range of behaviors discovered during training, which is especially clear when compared against SS, which differs only in the number of agents.

Overall, these results confirm the central hypothesis of this work: optimizing task-level objectives in a competitive manner not only waives the needs of dense behavioral rewards, but leads to overall better performance and flexibility.

D. Zero-Shot Deployment to the Real World

In this section, we evaluate how policies transfer to the real world. We test policies in a real-world replica of both the lemniscate and the complex track. We invite the reader to watch the supplementary video for more details.

We begin by comparing the sim-to-real transfer performance of the DS policy and our method. For each track layout, we conduct three head-to-head races, each consisting of three laps. As shown in Fig. 5, our approach achieves a 44.7% smaller gap between the average flight speed in simulation and in the real world (0.76 m/s for DS vs. 0.43 m/s for ours), along with a substantially lower real-world failure and collision rate (in simulation, rates are zero for both methods). These results indicate that competition with sparse rewards improves sim-to-real transfer.

Qualitatively, we observe the same trend as the track complexity increases: our method is the only one that successfully completes the lemniscate track in the presence of obstacles. We believe this improved sim-to-real transfer to be related to adversarial domain randomization [33], although we leave further analysis of this effect to future work.

To evaluate the real-world robustness of our approach against opponents it was not trained on, we conducted head-to-head competitions among all methods on the lemniscate track. Fig. 7 reports the results of these experiments, each conducted over three races. Our policy generalizes well, achieving the highest win rate (tied with DS). Additionally, in a real-world complex track race against DS, Ours won 2 out of 3 times, with one draw due to a collision.

These experiments reveal a subtle but interesting finding: Ours and DS have the same average win rates on the lemniscate track (Fig. 7), despite Ours winning direct head-to-head races. This is due to our approach having 3 draws with the DM baseline. Interestingly, this is not because the DM policy is particularly strong. Conversely, it did not transfer very well, and ended up consistently losing against all opponents. However, its behavior had an unexpected influence on our policy, making it crash twice, even if it had

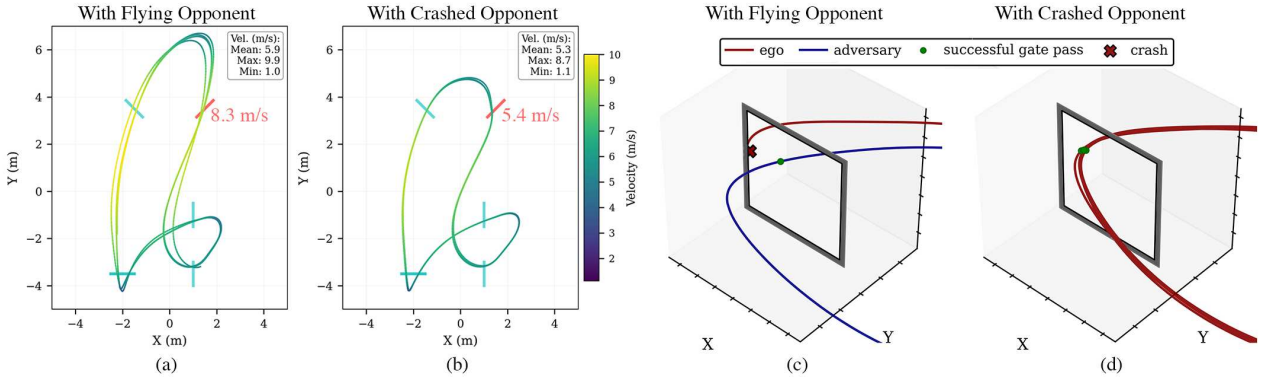

Fig. 6: (a, b) Agent trajectory and velocity profiles in races against an active adversary (not pictured) versus a crashed drone. Our policy exhibits risk-averse behavior once the opponent becomes non-competitive. (c) A learned blocking maneuver during a real-world race. (d) A real-world trajectory against a crashed opponent for comparison.

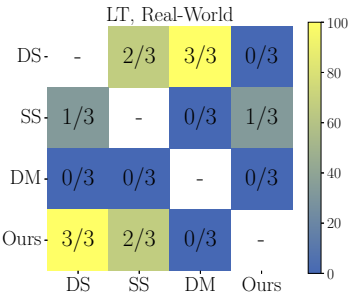

Fig. 7: Real-world race results on LT. DS and Ours achieve the same average win rate, but Ours directly outperforms DS. This is because DM transfers poorly, often exhibiting out-of-distribution behaviors that disrupt our policy.

Fig. 8: Cumulative reward for single-agent training. Dark curves represent mean rewards across three seeds. Training is stable and smooth, with very low variance.

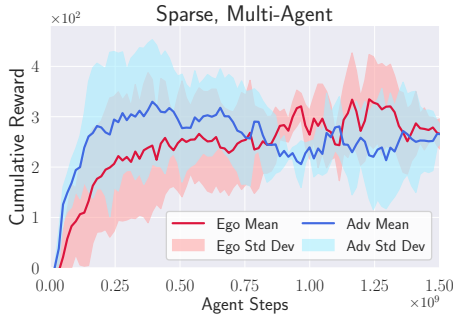

Fig. 9: Cumulative reward for multi-agent training, which exhibits greater variability, yet remains consistent overall.

a clear lead, and once at the beginning of the race, while both policies were rushing into the central gate. This highlights the challenges of simulation to reality transfer in multi-agent settings: the dependence on the behavior of another agent makes it challenging to predict the performance of a policy when deployed against new opponents. While a robust solution to this problem is outside of the scope of this work, we predict that large randomization at training time and adaptation at test time could alleviate these issues.

E. Emergence of strategic behaviors

To evaluate emerging strategic behaviors from sparse competitive rewards, we analyzed head-to-head self-play trajectories and velocities. On the Complex Track, our agent flies significantly more aggressively against competitive opponents than against crashed ones (Fig. 6 a-b). Specifically, against an active opponent, maximum and mean velocities reach 9.9 m/s and 5.9 m/s, but drops to 8.7 m/s and 5.3 m/s once the opponent crashes. We also observe that our agent maintains a substantially higher final-gate velocity of 8.3 m/s when the opponent is competitive, as opposed to 5.4 m/s when the opponent is crashed. This behavior is risk-averse, as the gate-pass rewards and lap bonus rewards become guaranteed once the opponent forfeits the racing task.

Blocking maneuvers are an important indicator of rich opponent-aware strategies emerging from sparse competitive multi-agent rewards. Fig. 6 (c and d) depicts one such maneuver in the real world. During competition, the adversary defends its marginal forward lead by flying a wide trajectory as it enters the gate (Fig. 6c), forcing the ego agent to the outside and causing a collision with the gate frame. In contrast, Fig. 6d illustrates the trajectory flown with a crashed opponent, which is clearly a safer trajectory.

In summary, the evidence from this section shows that our agent learns strategic behaviors beyond flying fast.

F. Training stability of multi-agent policies

Finally, we analyzed the training stability of multi-agent versus dense single-agent policies. Figs. 8 and 9 show the training curves for the DS policy and Ours (with seeds 1, 2,

and 3). As expected, the single-agent training is stable with very low variance. In contrast, multi-agent training exhibits greater variability but remains consistent overall, yielding highly competitive policies. Interestingly, the blue and red curves alternate in achieving higher cumulative rewards, reflecting the competitive nature of the multi-agent setup, as different phases emerge where one temporarily outperforms the other before the balance shifts again.

V. CONCLUSIONS

Our study provides evidence that agile and strategic low-level control in physical platforms can emerge from simple, sparse competition-based rewards. Moreover, it highlights their advantages over prescriptive terms that constrain agent behavior, both in terms of sim-to-real transfer and overall performance. However, this work only scratches the surface of what competitive rewards can achieve. Future extensions could explore scaling towards team-based competition, emergent active perception in vision-based agents, or robustness against rapidly adapting opponents.

VI. ACKNOWLEDGMENTS

This work was supported by the DARPA TIAMAT program (HR0011-24-9-0430). We thank the Google TPU Research Cloud Program, NSF ACCESS, and the National Center for Supercomputing Applications (NCSA) for computational resources. We also thank Himanshu Gaurav Singh, Pratik Kunapuli, and Chunwei Xing for helpful discussions and assistance throughout the project.

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] Y. Song, A. Romero, M. Mueller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, p. adg1462, 2023.

[3] 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, Aug 2023.

[4] R. Ferede, T. Blaha, E. Lucassen, C. De Wagter, and G. C. de Croon, "One net to rule them all: Domain randomization in quadcopter racing across different platforms," *arXiv preprint arXiv:2504.21586*, 2025.

[5] 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)*. IEEE, 2021, pp. 1205–1212.

[6] Y. Kang, J. Di, M. Li, Y. Zhao, and Y. Wang, "Autonomous multi-drone racing method based on deep reinforcement learning," *Science China Information Sciences*, vol. 67, no. 8, p. 180203, 2024.

[7] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, pp. 1–17, 2022.

[8] 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.

[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] R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager, "A game theoretic approach to autonomous two-player drone racing," *arXiv preprint arXiv:1801.02302*, 2018.

[11] M. Pustilnik, A. Loquercio, and F. Borrelli, "Non-normalized solutions of generalized nash equilibrium in autonomous racing," *arXiv preprint arXiv:2503.12002*, 2025.

[12] E. L. Zhu and F. Borrelli, "A sequential quadratic programming approach to the solution of open-loop generalized nash equilibria for autonomous racing," *arXiv preprint arXiv:2404.00186*, 2024.

[13] H. Lee, T. Seno, J. J. Tai, K. Subramanian, K. Kawamoto, P. Stone, and P. R. Wurman, "A champion-level vision-based reinforcement learning agent for competitive racing in gran turismo 7," *IEEE Robotics and Automation Letters*, 2025.

[14] K. Sims, "Evolving virtual creatures," in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '94, 1994, p. 15–22.

[15] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman *et al.*, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019.

[16] S. Liu, G. Lever, J. Merel, S. Tunyasuvunakool, N. Heess, and T. Graepel, "Emergent coordination through competition," *arXiv preprint arXiv:1902.07151*, 2019.

[17] M. F. Cusumano-Towner, D. Hafner, A. Hertzberg, B. Huval, A. Petrenko, E. Vinitsky, E. Wijmans, T. W. Killian, S. Bowers, O. Sener *et al.*, "Robust autonomy emerges from self-play," in *Forty-second International Conference on Machine Learning*, 2025.

[18] E. Vinitsky, R. Köster, J. P. Agapiou, E. A. Duéñez-Guzmán, A. S. Vezhnevets, and J. Z. Leibo, "A learning agent that acquires social norms from public sanctions in decentralized multi-agent settings," *Collective Intelligence*, vol. 2, no. 2, p. 26339137231162025, 2023.

[19] R. Ferede, C. De Wagter, D. Izzo, and G. C. de Croon, "End-to-end reinforcement learning for time-optimal quadcopter flight," *arXiv preprint arXiv:2311.16948*, 2023.

[20] R. Ferede, G. de Croon, C. De Wagter, and D. Izzo, "End-to-end neural network based optimal quadcopter control," *Robotics and Autonomous Systems*, vol. 172, p. 104588, 2024.

[21] D. Zhang, A. Loquercio, J. Tang, T.-H. Wang, J. Malik, and M. W. Mueller, "A learning-based quadcopter controller with extreme adaptation," *IEEE Transactions on Robotics*, 2025.

[22] F. Zhao, J. Mei, J. Zhou, Y. Chen, J. Chen, and S. Li, "Gate-aware online planning for two-player autonomous drone racing," *arXiv preprint arXiv:2402.18021*, 2024.

[23] R. Spica, E. Cristofalo, Z. Wang, E. Montijano, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1389–1403, 2020.

[24] J. Di, S. Chen, P. Li, X. Wang, H. Ji, and Y. Kang, "A cooperative-competitive strategy for autonomous multidrone racing," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 7, pp. 7488–7497, 2023.

[25] X. Wang, J. Zhou, Y. Feng, J. Mei, J. Chen, and S. Li, "Dashing for the golden snitch: Multi-drone time-optimal motion planning with multi-agent reinforcement learning," *arXiv preprint arXiv:2409.16720*, 2024.

[26] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[27] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent interaction," 2019.

[28] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *NeurIPS*, vol. 35, pp. 24 611–24 624, 2022.

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

[30] P. Kunapuli, J. Welde, D. Jayaraman, and V. Kumar, "Leveling the playing field: Carefully comparing classical and learned controllers for quadrotor trajectory tracking," 2025.

[31] J. Förster, "System identification of the crazyflie 2.0 nano quadrocopter," B.S. thesis, ETH Zurich, 2015.

[32] 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.

[33] R. Khirodkar and K. M. Kitani, "Adversarial domain randomization," *arXiv preprint arXiv:1812.00491*, 2018.