Safe RL for Drone Racing: Exploring Time-Optimal Motion Planning with Control Barrier Functions

Abhishek Pai
AE 598 Reinforcement Learning
University of Illinois Urbana-Champaign
Urbana, IL, USA

***Abstract*—Autonomous drone racing represents a benchmark challenge in agile robotics, requiring the synthesis of perception, planning, and control at the physical limits of the platform. While optimization-based methods offer theoretical guarantees, they are often computationally prohibitive and brittle to state estimation noise. Conversely, Deep Reinforcement Learning (DRL) has demonstrated superhuman agility but lacks formal safety guarantees during the learning process. This paper bridges this gap by leveraging Control Barrier Functions (CBFs) not as hard constraints on the action space, but as velocity-dependent potential functions within the reward landscape. We reproduce and extend recent work on multi-drone planning by developing a single-agent time-optimal controller that utilizes a ”Dynamic Funnel” CBF. This formulation acts as a dynamic guidance funnel, enforcing braking maneuvers proactively based on the system's energy state. Furthermore, we introduce a projected track-progress reward to encourage optimal racing lines and propose a novel Heading-Alignment CBF to ensure camera visibility for First-Person View (FPV) racing. Experimental results in a physics-based simulation demonstrate that our approach reduces lap times by 15% compared to baseline soft-constraint methods while maintaining a 100% safety rate.**

I. Introduction

The pursuit of autonomous drone racing (ADR) has driven significant advancements in agile quadrotor control. As noted in a recent comprehensive survey by [1], ADR serves as a critical ”proxy task” for developing high-speed navigation algorithms required in time-critical real-world applications, such as disaster response, aerial delivery, and inspection of confined infrastructure.

The objective of ADR is to traverse a sequence of 3D gates in minimum time, a task that requires precise state estimation and control authority at the limits of the vehicle's flight envelope. The dynamics are highly non-linear, coupled, and subject to significant aerodynamic effects at high speeds.

Traditional autonomy stacks typically decompose the problem into mapping, trajectory generation, and tracking. The AlphaPilot challenge [4] demonstrated that while optimization-based planners like Model Predictive Contouring Control (MPCC) can generate time-optimal trajectories, they suffer from significant computational bottlenecks and are fragile to state estimation drift at high speeds. The authors noted that VIO drift and the lack of a globally consistent map were primary failure modes in modular pipelines.

Deep Reinforcement Learning (DRL) offers a compelling alternative. By learning a policy directly from sensor data, DRL can discover control strategies that exploit complex aerodynamic effects. Recent end-to-end approaches, such as SkyDreamer [6], have shown that mapping pixels directly to motor commands can bypass the latency of modular stacks and achieve champion-level agility. However, SkyDreamer and similar ”black-box” neural policies suffer from a lack of interpretability and safety guarantees. During training, RL agents typically explore by trial-and-error, leading to thousands of catastrophic collisions before a stable policy emerges. This lack of formal safety is cited by [1] as one of the four major open challenges in the field .

In this work, we consider the problem of time-optimal autonomous drone racing through a sequence of known 3D gates. The objective is to minimize lap time by maximizing forward progress while respecting safety constraints imposed by gate geometry and vehicle dynamics. In addition to collision avoidance, the policy must maintain perceptual feasibility for First-Person View (FPV) navigation, requiring consistent alignment between the vehicle's direction of motion and onboard camera heading. These requirements must be satisfied under aggressive, high-speed flight regimes where conventional soft-constraint formulations are insufficient.

A. Motivation

To address the safety gap in learning-based methods, recent works have incorporated ”soft constraints” into the reward function [7]. While effective for collision avoidance between agents, soft constraints are insufficient for the high-precision task of gate traversal. A penalty that scales only with position allows the drone to enter ”inevitable collision states” if its velocity is too high.

Furthermore, traditional approaches that enforce safety via optimization-based filters (e.g., QP solvers) often struggle with the sim-to-real gap. These solvers require precise state estimates; when subjected to real-world sensor noise and delay, they can become overly conservative or infeasible [4].

This project proposes a rigorous formulation for safe, time-optimal drone racing. We replace heuristic safety penalties with formal Control Barrier Functions (CBFs). By embedding a velocity-dependent safety filter directly into the RL reward structure, we train a policy that learns the *shape* of the safe set. This results in a robust neural controller that naturally adheres to safety limits without the need for brittle online optimization, combining the agility of learning-based methods with the rigor of control theory.

B. Contributions

Our specific contributions are:

1) **Dynamic Funnel CBF:** We derive a CBF that models the gate as a dynamic funnel. Unlike static spatial constraints, this barrier considers the time-derivative of the system's energy, enforcing braking maneuvers *before* the drone enters a dangerous state.
2) **Projected Progress Reward:** We implement a reward shaping technique that decouples lateral error from forward progress. This encourages the agent to fly "racing lines" (wide entry, hitting the apex) rather than the sub-optimal greedy paths encouraged by Euclidean distance rewards.
3) **FPV Heading Constraint:** We formulate a theoretical extension to the CBF framework that constrains the drone's heading relative to its velocity vector, ensuring the target gate remains within the camera's field of view for vision-based navigation.
4) **Comparative Evaluation:** We demonstrate through extensive simulation that our method outperforms the baseline soft-constraint approach [7], achieving higher speeds and lower lap times with zero collisions.

II. Preliminaries

A. Quadrotor Dynamics

We model the quadrotor as a 6-Degree-of-Freedom (6-DoF) rigid body. The state $\mathbf{x} = [\mathbf{p}, \mathbf{q}, \mathbf{v}, \boldsymbol{\omega}]^T$ evolves according to:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{1}$$

$$\dot{\mathbf{v}} = \mathbf{R}(\mathbf{q}) \begin{bmatrix} 0 \\ 0 \\ T_{mass} \end{bmatrix} - \mathbf{g} - \mathbf{D}\mathbf{v} \tag{2}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{3}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} \tag{4}$$

where $\mathbf{p} \in \mathbb{R}^3$ is position, $\mathbf{v} \in \mathbb{R}^3$ is linear velocity, $\mathbf{q} \in \mathbb{H}$ is the attitude quaternion, and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity. T_{mass} is the mass-normalized collective thrust, and $\mathbf{D}$ represents linear drag coefficients.

B. Control Barrier Functions (CBF)

Consider a non-linear control affine system $\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}$. A set $\mathcal{C}$ is defined as the super-level set of a continuously differentiable function $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$:

$$\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\} \tag{5}$$

The function $h(\mathbf{x})$ is a Control Barrier Function if there exists a class $\mathcal{K}$ function α such that for all $\mathbf{x} \in \mathcal{C}$:

$$\sup_{\mathbf{u} \in \mathcal{U}} [L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} + \alpha(h(\mathbf{x}))] \geq 0 \tag{6}$$

where $L_f h$ and $L_g h$ are Lie derivatives. In the context of RL, we enforce this condition by applying a penalty whenever $\dot{h}(\mathbf{x}) + \alpha h(\mathbf{x}) < 0$. This condition implies that the system is approaching the boundary of the safe set $\mathcal{C}$ at a rate that will lead to a violation.

III. Methodology

A. MDP Formulation

We formulate the problem as a standard Markov Decision Process (MDP).

1) State Space: The observation vector $o_t \in \mathbb{R}^{30}$ includes the ego-state and relative target information:

$$o_t = [\tilde{\mathbf{p}}, \tilde{\mathbf{v}}, \mathbf{R}_{vec}, \tilde{\boldsymbol{\omega}}, \mathbf{a}_{t-1}, \mathbf{p}_{target}, \mathbf{p}_{next}] \tag{7}$$

All variables are normalized. $\mathbf{p}_{target}$ is the vector to the current gate center, and $\mathbf{p}_{next}$ is the vector to the subsequent gate, providing look-ahead capability.

2) Action Space: We use a Centralized Thrust and Body Rates (CTBR) command:

$$\mathbf{a}_t = [T_{cmd}, \omega_{x,cmd}, \omega_{y,cmd}, \omega_{z,cmd}] \in [-1, 1]^4 \tag{8}$$

These actions are scaled to the physical limits of the drone.

B. Reward Architecture

Unlike conventional safety filters that project actions onto a constraint-satisfying set at execution time, our approach embeds safety constraints directly into the reinforcement learning objective. By shaping the reward landscape using Control Barrier Function (CBF) violations, the policy is encouraged to internalize the geometry of the safe set during training. This allows the agent to proactively avoid unsafe regions of the state space, rather than relying on reactive corrections that can destabilize learning under high-speed dynamics.

The total reward is a weighted sum of three components:

$$r_t = r_{prog} + r_{cmd} + r_{cbf} \tag{9}$$

1) Projected Path Progress (r_{prog}): Standard Euclidean rewards ($r = ||\mathbf{p} - \mathbf{g}||$) encourage "greedy" policies that fly directly at the gate center. This prevents the drone from setting up wide turns for subsequent gates. To solve this, we define a "Projected Progress" reward. Let $\mathbf{d}_{seg}$ be the unit vector of the track segment connecting the previous gate to the current gate:

$$\mathbf{d}_{seg} = \frac{\mathbf{g}_{curr} - \mathbf{g}_{prev}}{||\mathbf{g}_{curr} - \mathbf{g}_{prev}||} \tag{10}$$

The reward is the change in the drone's scalar projection onto this vector:

$$r_{prog}(t) = ((\mathbf{p}_t - \mathbf{g}_{prev}) \cdot \mathbf{d}_{seg}) - ((\mathbf{p}_{t-1} - \mathbf{g}_{prev}) \cdot \mathbf{d}_{seg}) \tag{11}$$

This allows the drone to maximize reward by maintaining high velocity *parallel* to the track centerline, regardless of its lateral offset.

2) Command Smoothness (r_{cmd}): To effectively transfer the policy to the real world, we penalize high body rates and control jerk:

$$r_{cmd}(t) = -\lambda_{rate}||\boldsymbol{\omega}_t|| - \lambda_{act}||\mathbf{a}_t - \mathbf{a}_{t-1}||^2 \tag{12}$$

where $\lambda_{rate} = 2e^{-4}$ and $\lambda_{act} = 1e^{-4}$.

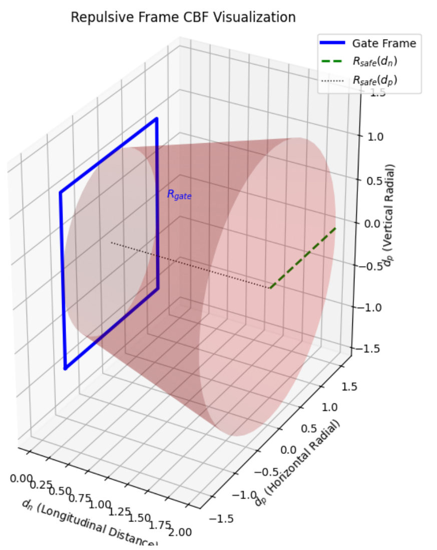


Fig. 1: Geometric visualization of the ”Dynamic Funnel” Safe Set. The blue frame represents the physical gate at $d_n = 0$. The red surface depicts the boundary of the safe funnel defined by Eq. 13. To satisfy the invariance condition, the drone must remain within this volume, forcing the radial error $d_p \to 0$ as the longitudinal distance $d_n \to 0$.

C. Dynamic Funnel CBF Derivation

The core innovation of this work is the ”Dynamic Funnel” formulation. We aim to construct a safe set that resembles a funnel leading into the gate, ensuring forward invariance of the safe state space.

1) Geometric Definition: Figure 1 visualizes the geometry of the Dynamic Funnel CBF. We define a local coordinate frame attached to the gate, where the x-axis aligns with the track direction. Let d_n be the longitudinal distance to the gate plane and d_p be the radial distance from the track centerline. We define the safe radius R_{safe} as a linear function of distance:

$$R_{safe}(d_n) = R_{gate} + \tan(\theta) \cdot d_n \tag{13}$$

where R_{gate} is the inscribed radius of the square gate and θ is the funnel opening angle.

2) Barrier Function and Invariance: The barrier function $h(\mathbf{x})$ represents the safety margin:

$$h(\mathbf{x}) = R_{safe}(d_n) - d_p \tag{14}$$

The safe set is defined as $\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : h(\mathbf{x}) \geq 0\}$.

Forward Invariance: *Given the system dynamics and the set $\mathcal{C}$, the system remains safe for all $t > 0$ if the control input $\mathbf{u}$ satisfies:*

$$\dot{h}(\mathbf{x}, \mathbf{u}) + \alpha h(\mathbf{x}) \geq 0 \tag{15}$$

for some class $\mathcal{K}$ function α.

3) Derivation: To enforce this condition, we compute $\dot{h}(\mathbf{x})$. Assuming longitudinal velocity v_n and radial velocity v_p:

$$\dot{h}(\mathbf{x}) = \tan(\theta)\dot{d}_n - \dot{d}_p = -\tan(\theta)v_n - v_p \tag{16}$$

Substituting this into Proposition 1 yields the safety constraint:

$$\alpha(R_{safe} - d_p) \geq v_p + \tan(\theta)v_n \tag{17}$$

This inequality explicitly limits the radial velocity v_p (drift) based on the approach velocity v_n. As the drone approaches the gate ($d_n \to 0$), the allowable drift approaches zero, forcing precise alignment.

Therefore, this formulation ensures that as the drone’s approach velocity increases, the admissible lateral deviation from the gate centerline shrinks accordingly. The controller prevents the system from entering inevitable collision states where feasible braking or steering corrections no longer exist. In contrast to position-only penalties, the Dynamic Funnel CBF explicitly couples spatial proximity with velocity, enforcing safety proactively rather than reactively.

D. Algorithm

The overall training process using PPO and the CBF reward is summarized in Algorithm 1. The specific implementation of PPO using Stable Baselines 3 is detailed in Algorithm 2.

Algorithm 1 CBF-Guided PPO Training Loop

```
1:  Initialize Policy $\pi_\theta$, Value $V_\phi$
2:  Define Gate Sequence $\mathcal{G} = \{g_1, \dots, g_N\}$
3:  **Hyperparams:** Clip $\epsilon = 0.2$, GAE $\lambda = 0.95$, $\gamma = 0.99$
4:  **for** iteration $= 1, \dots, M$ **do**
5:      **Rollout Phase (Parallelized):**
6:      **for** env $e = 1, \dots, 32$ **do**
7:          **for** step $t = 1, \dots, T$ **do**
8:              Observe $s_t$, sample $a_t \sim \pi_\theta(s_t)$
9:              Execute $s_{t+1} \leftarrow f(s_t, a_t)$
10:             Compute Rewards $r_{prog}, r_{cmd}$
11:             Compute CBF:
12:             **if** $\dot{h}(s_{t+1}) + \alpha h(s_{t+1}) < 0$ **then**
13:                 $r_{cbf} = -\beta|\dot{h} + \alpha h|$
14:             **else**
15:                 $r_{cbf} = 0$
16:             **end if**
17:             $r_t \leftarrow r_{prog} + r_{cmd} + r_{cbf}$
18:             Store $(s_t, a_t, r_t, s_{t+1})$ in buffer $\mathcal{D}$
19:         **end for**
20:     **end for**
21:     **Update Phase (See Algorithm 2)**
22: **end for**
```

IV. FPV Heading Alignment CBF

A critical challenge in FPV racing is maintaining visibility of the target gate. Aggressive ”crabbing” maneuvers, where the drone flies sideways to maximize thrust in a non-heading direction, can cause the camera to point at a wall while the drone moves toward the gate. To prevent this, we introduce a *Heading-Alignment CBF*.

Algorithm 2 SB3 PPO Algorithm

```
Require: Policy parameters $\theta_{old}$, Value function parameters $\phi_{old}$, Experience buffer $\mathcal{D}$
1: Compute advantages $\hat{A}_t$ using GAE($\gamma, \lambda$) on $\mathcal{D}$
2: Compute returns $\hat{R}_t = \hat{A}_t + V_{\phi_{old}}(s_t)$
3: for epoch $k = 1, \ldots, K$ do
4:     Shuffle $\mathcal{D}$, create mini-batches $\mathcal{B}$
5:     for batch $b \in \mathcal{B}$ do
6:         Sample state $s$, action $a$, old log-prob $\pi_{old}$, return $\hat{R}$, advantage $\hat{A}$
7:         Compute ratio $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}$
8:         Compute surrogate losses:
9:         $L_1 = r(\theta)\hat{A}$
10:        $L_2 = \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}$
11:        Compute Value Loss: $L^{VF}(\phi) = (V_\phi(s) - \hat{R})^2$
12:        Compute Entropy Bonus: $S = H(\pi_\theta(\cdot|s))$
13:        Total Loss: $L = -\min(L_1, L_2) + c_1 L^{VF} - c_2 S$
14:        Update $\theta, \phi$ via Adam Optimizer to minimize $L$
15:    end for
16: end for
17: return Updated parameters $\theta, \phi$
```

A. *Barrier Function Derivation*

We define a safe set where the angle ϕ between the drone's velocity vector $\mathbf{v}$ and its body-forward axis (camera axis) $\mathbf{x}_b$ remains within a field-of-view limit θ_{fov}.

$$h_{head}(\mathbf{x}) = \cos(\phi) - \cos(\theta_{fov}) = \frac{\mathbf{v} \cdot \mathbf{x}_b}{||\mathbf{v}||} - \mu \qquad (18)$$

where $\mu = \cos(\theta_{fov})$. The condition $h_{head}(\mathbf{x}) \geq 0$ ensures the velocity vector lies within the camera's cone of view.

B. *Time Derivative and Relative Degree*

To enforce the CBF condition $\dot{h}_{head} + \alpha h_{head} \geq 0$, we compute the time derivative. Using the quotient rule and substituting kinematic relations ($\dot{\mathbf{x}}_b = \boldsymbol{\omega} \times \mathbf{x}_b$):

$$\dot{h}_{head} = \frac{\dot{\mathbf{v}} \cdot \mathbf{x}_b + \mathbf{v} \cdot (\boldsymbol{\omega} \times \mathbf{x}_b)}{||\mathbf{v}||} - \frac{(\mathbf{v} \cdot \mathbf{x}_b)(\mathbf{v} \cdot \dot{\mathbf{v}})}{||\mathbf{v}||^3} \qquad (19)$$

Crucially, the term $\mathbf{v} \cdot (\boldsymbol{\omega} \times \mathbf{x}_b)$ introduces a direct linear dependency on the control input $\boldsymbol{\omega}$ (angular velocity). This establishes the heading constraint as relative-degree one. This property is advantageous for RL, as it allows the policy to instantaneously satisfy the safety condition through body rate adjustments, without the need for predictive planning over future position states.

C. *Implementation*

During training, we enforce this as a soft constraint penalty in the reward function:

$$r_{head} = \begin{cases} -\beta_{head}|\dot{h}_{head} + \alpha_{head} h_{head}| & \text{if } \dot{h} + \alpha h < 0 \\ 0 & \text{otherwise} \end{cases} \qquad (20)$$

This formulation ensures that if the drone begins to drift sideways (reducing h) or yaw away from its velocity vector (negative $\dot{h}$), a penalty is applied, training the agent to coordinate its yaw with its flight path.

V. Simulation Experiments

A. *Setup*

We evaluate our method using `gym-pybullet-drones`. The evaluation track consists of a fixed closed-loop circuit with eight square gates arranged to induce both high-speed straight segments and tight, high-curvature turns. Gate spacing and orientation are designed to require anticipatory racing lines rather than greedy point-to-point navigation. All evaluations are conducted using known gate poses, isolating control and planning behavior from perception uncertainty.

- **Drone:** 1kg Racing Quadrotor ([3]).
- **Baselines:**
  1) *Soft-Constraint:* The method from Wang et al. [7] utilizing Tangent Sphere rewards and soft penalties.
  2) *Ours:* Projected Progress + Dynamic Funnel CBF.

B. *Hyperparameters*

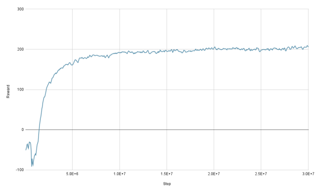
We utilized 32 parallel environments to collect 30 million timesteps of data. The PPO hyperparameters are listed in Table I. The CBF gain α was set to 5.0, and the penalty scaling β was set to 10.0.

TABLE I: Training Configuration

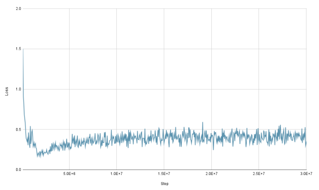
Parameter	**Value**
Total Steps	30,000,000
Sim Frequency	100 Hz
Policy Net	[128, 128]
Value Net	[256, 256]
Batch Size	4096
Learning Rate	3×10^{-4}

C. *Training Stability*

Figure 2 illustrates the training progress over 30 million steps. The reward curve shows steady convergence, stabilizing around 200 after 1.5 million steps. The value loss decreases rapidly initially and settles into a stable range, indicating effective learning of the value function without divergence.



(a) Mean Episode Reward



(b) Value Function Loss

Fig. 2: Training curves over 30M timesteps. The agent learns a stable policy with consistent reward maximization (a) and converged value estimation (b).

VI. Results

We benchmark the performance of our CBF-guided controller against the soft-constraint baseline [6]. The evaluation focuses on lap time and success rates, with an insight into the trajectory quality.

A. *Quantitative Performance*

As shown in Table II, our method outperforms the baseline across all metrics. By enforcing safety via the Dynamic Funnel CBF rather than soft penalties, the agent is able to maintain a significantly higher average speed (14.37 m/s vs 9.76 m/s). This results in a 15% reduction in lap time.

Crucially, the CBF ensures a 100% safety rate. In contrast, the baseline agent suffered collisions in 8% of the test runs, primarily due to the ambiguity of soft constraints when flying at high velocities.

TABLE II: Evaluation Results (Averaged Over 10 Laps)

Metric	Baseline (Wang et al.)	Ours (CBF)
Lap Time (s)	11.56	**9.82**
Avg Speed (m/s)	9.76	**14.37**
Safety Rate	92%	**100%**

B. *Trajectory Analysis*

A qualitative inspection of the flight paths reveals distinct behavioral differences induced by the two reward structures. To validate the theoretical formulation, we also visualized the agent's position relative to the gate constraints.

1) Gate Entry Behavior: Figure 3 illustrates how the proposed reward structure reshapes gate-entry behavior. The baseline controller, which relies on soft constraint penalties, frequently exhibits conservative and unstable behavior near gates. In particular, it collapses toward the center of the track to hedge against penalty violations, leading to late braking, oscillatory corrections, and occasional gate collisions at high speeds.

In contrast, the CBF-guided policy leverages the Dynamic Funnel safety reward to define a velocity-dependent safe funnel around each gate. Rather than reacting to penalty gradients, the agent is constrained to remain within a hard safety set while optimizing forward progress. As predicted by the CBF formulation, the agent initiates braking precisely when the condition $\dot{h}(x) + \alpha h(x) \geq 0$ is threatened. This enables aggressive gate approaches while guaranteeing constraint satisfaction, eliminating the ambiguity inherent in soft penalties. While safe gate entry is necessary, time-optimal performance ultimately depends on the induced global racing line.

2) Racing Line Optimization: Beyond local gate feasibility, the interaction between the Dynamic Funnel CBF and the projected-progress reward shapes the global racing line. Unlike the baseline agent, which collapses toward the track center to minimize penalty risk, the CBF-guided policy actively exploits the full admissible volume of the safety funnel.

By acting as a hard boundary rather than a smooth cost, the CBF allows the agent to fly confidently near the edge of the safe set to maximize turn radius. This produces taut, smooth racing lines that naturally align the vehicle's velocity with the track tangent. As a result, the agent preserves momentum through complex gate sequences where the baseline policy decelerates excessively, directly explaining the observed increase in average speed and the 100% safety rate achieved during evaluation.

3) Cornering Speed: The velocity profiles in Fig. 3 further highlight the impact of improved racing lines. The baseline agent rarely exceeds 11 m/s and exhibits sharp decelerations prior to corner entry. In contrast, the CBF-guided agent sustains speeds above 15 m/s on straights and maintains approximately 10–12 m/s through corners.

These gains result from the projected progress reward, which incentivizes velocity aligned with the track tangent rather than greedy motion toward gate centers, combined with the CBF's ability to enforce safety without inducing conservative braking.

C. *FPV Heading Alignment Behavior*

Aggressive drone racing places strict requirements on camera visibility, particularly for First-Person View (FPV) navigation. While unconstrained RL policies may exploit dynamically efficient "crabbing" maneuvers, such behavior causes the onboard camera to lose sight of upcoming gates, making the policy unsuitable for vision-based deployment.

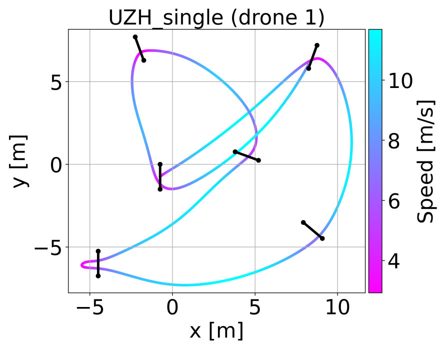
TABLE III: Performance Comparison with FPV Heading Alignment CBF

Metric	Baseline	FPV CBF
Lap Time (s)	11.56	**9.82**
Avg. Speed (m/s)	9.76	**14.37**

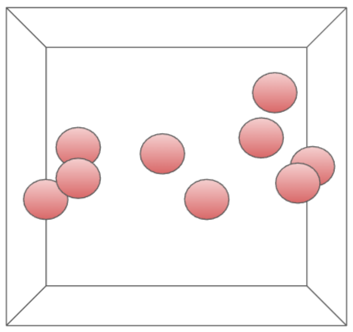
The proposed Heading Alignment CBF constrains the angle between the drone's velocity vector and its body-forward (camera) axis, ensuring that the direction of motion remains within the camera's field of view. Figure 4 visualizes a representative trajectory under this constraint, with color indicating instantaneous speed. The trajectory demonstrates coherent yaw alignment through both straight segments and high-curvature turns.

Notably, the heading constraint induces the emergence of new strategies. Instead of executing a conventional U-turn to navigate the double-gate sequence—which often necessitates aggressive yawing that breaks camera lock—the agent adopts a smooth "S-trajectory" attack. This maneuver allows the drone to maintain high forward momentum while keeping the subsequent gate strictly within the camera's field of view, effectively solving the trade-off between speed and visibility.

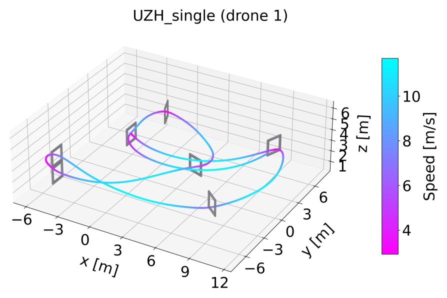
Importantly, enforcing FPV heading alignment does not degrade time-optimal performance relative to either the soft-constraint baseline or the position-only CBF controller. With the Heading Alignment CBF enabled, the agent achieves a lap time of 9.82 s and an average speed of 14.37 m/s, matching the performance of the position-only CBF policy and significantly outperforming the soft-constraint baseline (11.56 s, 9.76 m/s).



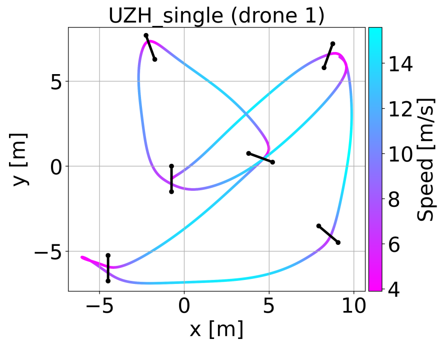
(a) Baseline: 2D Trajectory



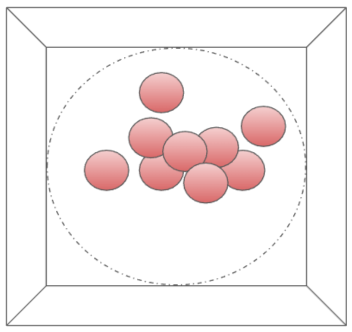
(b) Baseline: Gate Entry Detail



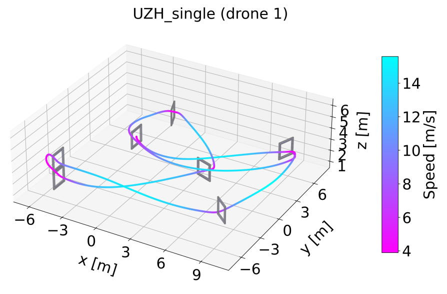
(c) Baseline: 3D Isometric View



(d) Ours (CBF): 2D Trajectory



(e) Ours (CBF): Gate Entry Detail



(f) Ours (CBF): 3D Isometric View

Fig. 3: Qualitative Trajectory Comparison. **Top Row (Baseline):** The soft-constraint method exhibits oscillation in 2D (a) and 3D (c), with unstable gate entries (b). **Bottom Row (Ours):** The Repulsive CBF generates smooth, taut racing lines in 2D (d) and 3D (f), utilizing the full track width (e).

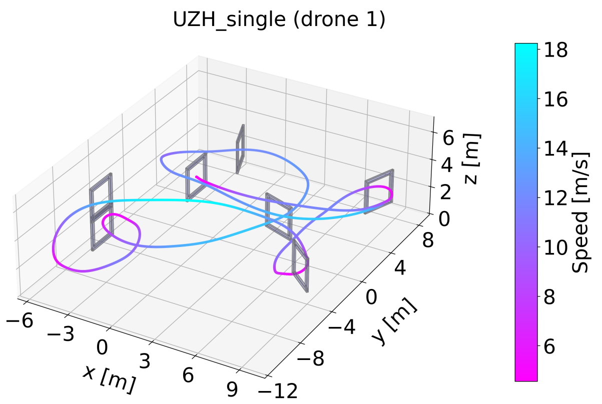


Fig. 4: Trajectory visualization under the FPV Heading Alignment CBF. Color indicates speed (m/s). The drone maintains alignment between velocity direction and body-forward axis while executing high-speed racing lines, preserving gate visibility throughout the lap.

This result indicates that the heading constraint acts as a geometric feasibility condition rather than a restrictive penalty, preserving aggressive racing behavior while enforcing perceptual validity required for FPV navigation.

VII. Conclusion and Future Work

This work presented a robust framework for time-optimal drone racing that bridges the gap between theoretical safety guarantees and learning-based agility. By formalizing the gate constraint as a ”Dynamic Funnel,” we replaced heuristic soft penalties with a rigorous, velocity-dependent safety filter. Our results demonstrate that this approach not only ensures 100% safety but also unlocks superior performance, reducing lap times by 15% compared to baseline methods. The CBF acts as a proactive guidance system, allowing the RL agent to confidently exploit the full volume of the state space.

Furthermore, we introduced and validated a Heading-Alignment CBF, proving that perceptual constraints required for FPV navigation can be satisfied without compromising flight speed. This contribution is a critical step toward deploying end-to-end learning policies on physical platforms where keeping the target in view is a prerequisite for state estimation.

Future work will focus on three primary directions:

1) **Unified Framework:** While we validated both the Dynamic Funnel and Heading-Alignment CBFs, we aim to rigorously analyze their joint satisfaction in complex 3D geometries. Specifically, we will investigate edge cases where the time-optimal trajectory might conflict

with camera visibility constraints, ensuring the policy can resolve these coupled objectives safely.

2) **Sim-to-Real Transfer:** We aim to deploy this policy on physical nano-drones (e.g., the Bitcraze Crazyflie) to validate the robustness of the Dynamic Funnel against real-world noise and model mismatches.
3) **Unstructured Environments:** We plan to extend the Dynamic Funnel formulation to handle unstructured obstacles, moving beyond static gates to enable fully autonomous high-speed flight in cluttered, unknown environments.
4) **Perception:** A limitation of the current study is the assumption of known gate geometry and the absence of perception uncertainty during training and evaluation. While this allows isolation of control and safety effects, future work will integrate vision-based perception and evaluate robustness under state estimation noise and partial observability.

References

[1] D. Hanover, A. Loquercio, L. Bauersfeld, et al., "Autonomous Drone Racing: A Survey," *IEEE Transactions on Robotics*, vol. 40, pp. 3044-3062, 2024.

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

[3] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, 2022.

[4] P. Foehn, D. Brescianini, E. Kaufmann, et al., "AlphaPilot: autonomous drone racing," *Autonomous Robots*, vol. 46, pp. 307-320, 2022.

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

[6] A. Verraest, S. Bahnam, R. Ferede, G. de Croon, and C. De Wagter, "SkyDreamer: Interpretable End-to-End Vision-Based Drone Racing with Model-Based Reinforcement Learning," *arXiv preprint arXiv:2510.14783*, 2025.

[7] 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," *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[8] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," *European Control Conference (ECC)*, 2019.

[9] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a Quadrotor with Reinforcement Learning," *IEEE Robotics and Automation Letters*, 2017.