Vision Based Drone Flight: Team 4 Report

Jianwen Cao
Dept. Informatics
University of Zurich
Zurich, Switzerland
jianwen.cao@uzh.ch

Shaurya Kishore Panwar
Dept. Informatics
University of Zurich
Zurich, Switzerland
shauryakishore.panwar@uzh.ch

Rui Zhou
Dept. Informatics
University of Zurich
Zurich, Switzerland
rui.zhou2@uzh.ch

Zizhou Luo
Dept. Informatics
University of Zurich
Zurich, Switzerland
zizhou.luo@uzh.ch

Kartikeya Dubey
Dept. Informatics
University of Zurich
Zurich, Switzerland
kartikeya.dubey@uzh.ch

***Abstract*—This project addresses the challenge of autonomous vision-based drone flight, focusing on Task 2: developing a reinforcement learning (RL) controller for a camera drone to track a moving actor drone while maintaining it within the onboard camera's field of view (FoV) and avoiding collisions in a constrained flight volume. The controller must process real-time inputs, including the camera drone's position, attitude, velocity, body rates, and the actor drone's estimated position, to output safe control commands. Our methodology unfolds in two stages: a baseline state-based controller using ground-truth actor positions, and an advanced vision-based controller relying on bounding box detections from a double-sphere camera model. The state-based policy, implemented as a neural network trained via proximal policy optimization (PPO) [1], successfully learned stable tracking behaviors in simulation, demonstrating robust performance under idealized conditions with low tracking error and high FoV retention. However, the vision-based extension proved more challenging due to noisy bounding box inputs and partial observability. Despite iterative training on the Agilicious simulator, the policy did not converge, exhibiting oscillatory and unstable flights. This report details the policy architectures, training pipelines, and simulation results for both approaches, followed by a discussion of encountered challenges—such as reward formulation and sim-to-real gaps—and avenues for improvement, including curriculum learning and domain randomization. The recording of our demo can be found in https://www.youtube.com/watch?v=H1y3LKRlMYc.**

***Index Terms*—Reinforcement Learning, Autonomous Drones, Proximal Policy Optimization (PPO), Sim-to-Real Transfer**

I. Introduction

Autonomous vision-based drone flight is a rapidly advancing field with transformative applications, including aerial cinematography, infrastructure inspection, search-and-rescue operations, and human–robot collaboration. A foundational capability for these scenarios is *target following*: a camera-equipped drone must robustly track a dynamic actor—such as a moving drone or object-of-interest—while maintaining it within the onboard camera's field of view (FoV), adhering to safe standoff distances, and avoiding collisions within a bounded flight volume. This work addresses this challenge by formulating target following as a reinforcement learning (RL) control problem under perceptual noise and partial observability, evaluating policies in both simulation and real-world deployments.

Traditional feedback controllers, such as proportional–integral–derivative (PID) and model predictive control (MPC), excel in static or well-modeled environments but falter amid rapid dynamics, unmodeled disturbances, and sensor uncertainties. These limitations often yield fragile policies confined to narrow operating zones. In contrast, RL enables an agent to learn adaptive control policies through trial-and-error interactions, optimizing a task-specific reward without explicit dynamics models. This approach fosters emergent behaviors, such as maintaining an optimal orbital trajectory around the target for sustained visual contact, and accommodates heterogeneous motion patterns and sensing degradations.

We study a *camera drone* (the RL agent) tasked with tracking a *moving actor drone*. At each timestep, the agent observes its ego-state—3D position $\mathbf{p}_d \in \mathbb{R}^3$, rotation matrix $\mathbf{R}_d \in \mathbb{R}^{3\times3}$, linear velocity $\mathbf{v}_d \in \mathbb{R}^3$, and angular rates $\boldsymbol{\omega}_d \in \mathbb{R}^3$—along with either (i) the actor's ground-truth state $(\mathbf{p}_a, \mathbf{R}_a, \mathbf{v}_a, \boldsymbol{\omega}_a)$ (state-based variant) or (ii) a noisy 2D bounding box detection in normalized image coordinates (vision-based variant, rendered assuming a double-sphere camera model). The policy outputs normalized commands for collective thrust T and angular rates $(\omega_x, \omega_y, \omega_z)$, denormalized to physical limits (e.g., $T \in [0, 20]$ m/s^2, rates $\in [-10, 10]$ rad/s). The objective is to minimize tracking error, center the actor in the FoV, and ensure collision-free flight, encapsulating the interplay of pursuit, visual servoing, and safety under scale ambiguity and observation noise.

To reduce complexity and risks, we advance through three progressive stages:

1) **Hovering and trajectory tracking.** A foundational policy learns point-to-point navigation and stable hovering at arbitrary points, extended to follow prescribed trajectories—including a real-world deployment demonstrating adherence without manual intervention.
2) **State-based target following.** Assuming perfect actor localization, we train a proximal policy optimization (PPO) [1] agent in the Agilicious simulator. This yields reliable tracking with sub-meter position errors and $> 95\%$ FoV retention, validating reward designs for proximity, stability, and smoothness while establishing an idealized baseline.
3) **Vision-based target following.** Ground-truth access is replaced by simulated bounding boxes, introducing partial observability and noise. Despite extended training (2M steps with noise curriculum), convergence remains elusive, manifesting in oscillatory pursuits and frequent

FoV losses—exposing bottlenecks in perceptual inference and motion prediction.

This incremental methodology has two purposes- it separates RL's strengths (e.g., adaptive stabilization around dynamic targets) from perceptual hurdles (e.g., 3D intent derivation from 2D cues), and it delivers deployable intermediates (e.g., trajectory followers) to judge progress. We dissect pivotal design elements—reward shaping to balance tracking , FoV compliance, and control regularization.

The challenges posed by the transition to a vision-based model, such as perceptual aliasing and delayed detection, underscore needs for enhancements like progressive curricula (ramping target speeds and noise levels) and domain randomization (varying lighting and occlusions). These insights guide practical steps, detailed below, to bridge the sim-to-real gap.

In essence, this study probes RL-driven target following across sensing types, from ground-truth state availability to vision-only inputs. The state-based policy attains simulation benchmarks and real trajectory fidelity, whereas the vision extension showcases observability constraints. Our analysis pinpoints points of failure and charts potential solutions towards resilient, deployable vision-based drone autonomy.

II. Methodology

We followed the structured pipeline for Task 2, implementing a reinforcement learning (RL) controller within the provided codebase. This environment models the dynamics of a camera drone tracking an actor drone. We developed both a state-based baseline and a vision-based policy, training each using a customized Proximal Policy Optimization (PPO) algorithm [1] from Stable Baselines 3, with modifications to reward structures and observation spaces to handle increasing perceptual complexity.

A. State-Based Controller

The state-based controller leverages ground-truth actor positions from motion capture, establishing a fully observable baseline to validate the RL framework. This setup ensures stable tracking under idealized conditions, isolating vision-related challenges for the later steps.

Observation Space: A 24D vector comprising:

- Camera drone: 3D position $\mathbf{p}_d \in \mathbb{R}^3$, 6D rotation matrix $\mathbf{R}_d \in \mathbb{R}^{3\times3}$, 3D linear velocity $\mathbf{v}_d \in \mathbb{R}^3$, and 3D angular velocity $\boldsymbol{\omega}_d \in \mathbb{R}^3$.
- Actor: Current 3D position $\mathbf{p}_a \in \mathbb{R}^3$, previous position $\mathbf{p}_{a,\text{prev}} \in \mathbb{R}^3$, and two-steps-prior position $\mathbf{p}_{a,\text{prev2}} \in \mathbb{R}^3$.

The inclusion of temporal position history (current, previous, and two-steps-prior) enhances motion prediction, enabling the policy to anticipate actor dynamics. Direct access to precise state data minimizes latency and ensures robust tracking, serving as a foundation for policy creation.

Initialization: To bridge the sim-to-real gap, the environment employs randomized initialization, placing the camera drone at the arena's center and perturbing its state with controlled noise (position: full randomization; orientation, velocity, and angular rates: scaled by 0.25, 0.2, 0.2, respectively). The actor's trajectory is generated via a RandomTrajectory generator within 80% of the workspace bounds, seeded uniquely per episode. This randomization adds robustness to initial conditions and target motion, mimicking real-world variability and improving policy generalizability.

Action Space: A 4D continuous $\texttt{Box}(-1, 1)$ vector: normalized collective thrust T and angular rates $(\omega_x, \omega_y, \omega_z)$. Actions are denormalized via $\pi_{\text{act}} = \text{act} \cdot \text{act\_std} + \text{act\_mean}$ and clipped to $[\text{act\_min}, \text{act\_max}]$, yielding thrust in $[0, 20]$ m/s^2 and rates in $[-10, 10]$ rad/s. This design ensures bounded, smooth control inputs, facilitating stable exploration.

Neural Network Architecture (Actor-Critic Policy): A shared 2-layer MLP feature extractor (256 and 128 units, ReLU activations) feeds into an actor head (linear layer with Tanh for bounded outputs) and a critic head (linear layer for value estimation). Orthogonal weight initialization mitigates vanishing gradients, enhancing training stability for efficient convergence.

Reward Function: The reward function, aligned with the provided C++ implementation, is designed to be gradient-friendly, prioritizing proximity and stability via penalties scaled by distance to target. This structure accelerates PPO convergence by providing dense feedback while balancing tracking accuracy and flight smoothness.

$$r_{\text{track}} = \exp\left(-\frac{(d_{xy} - 1.5)^2}{2}\right) + 0.5, \tag{1}$$

A Gaussian reward encouraging an ideal XY-plane distance $d_{xy} = \|\mathbf{p}_d^{xy} - \mathbf{p}_a^{xy}\|$ of 1.5 m, with a $+0.5$ offset to maintain positive rewards and support early exploration.

$$r_z = \exp(-d_z^2), \tag{2}$$

An exponential penalty on altitude difference $d_z = |\mathbf{p}_d^z - \mathbf{p}_a^z|$, promoting vertical alignment for stable tracking.

$$r_{\text{vel}} = -0.05 \cdot \exp(-d_{3D}) \cdot \|\mathbf{v}_d\|, \tag{3}$$

a distance-gated penalty ($d_{3D} = \|\mathbf{p}_d - \mathbf{p}_a\|$) on linear velocity magnitude, encouraging hovering near the actor to dampen oscillations.

$$r_{\text{ang}} = -0.05 \cdot \exp(-d_{3D}) \cdot \|\boldsymbol{\omega}_d\|, \tag{4}$$

a similar penalty on angular velocity magnitude, reducing erratic rotations for smoother control (coefficient 0.05 balances stability and responsiveness).

$$r_{\text{act}} = -0.05 \cdot \exp(-d_{3D}) \cdot \left(\|\pi_{\text{act}}[1:4]\| + |T - 9.81|\right), \tag{5}$$

regularizing control inputs by penalizing angular rates $\pi_{\text{act}}[1:4]$ and deviations from hover thrust (9.81 m/s^2), promoting energy-efficient, stable maneuvers.

The total reward is

$$r = r_{\text{track}} + r_z + r_{\text{vel}} + r_{\text{ang}} + r_{\text{act}}, \tag{6}$$

with unit weights. Notably, the orientation reward r_{ori} is disabled ($r_{\text{ori}} = 0$) in the provided code, as upright posture is less critical under full observability, simplifying the reward structure without sacrificing performance. Distance gating in (3), (4), and (5) focuses penalties on close-range interactions, enhancing learning efficiency by prioritizing relevant states.

B. Vision-Based Controller

The vision-based controller transitions to partial observability by replacing ground-truth positions with simulated bounding box detections, mimicking real-world sensor noise and occlusions. Key differences include a 27D observation space with temporal bounding box sequences for motion prediction, vision-specific reward terms (e.g., bounding box centering), and a target-lost penalty (-5) absent in the state-based design. Training extends to 2M timesteps with a noise curriculum (versus 1M without noise), addressing perceptual uncertainties while maintaining compatibility with the state-based framework for direct comparison.

Observation Space: A 27D vector extending the 18D drone state ($\mathbf{p}_d, \mathbf{R}_d, \mathbf{v}_d, \boldsymbol{\omega}_d$) with:

- Actor: Current (4D: $[x_{\min}, y_{\min}, w, h]$ in normalized $[-1, 1]$ image coordinates), previous, and two-steps-prior bounding boxes.

The temporal sequence mitigates noise and occlusion effects, initializing as zeros and concatenated segment-wise, trading precision for robustness critical for real-world deployment.

Initialization To enhance sim-to-real transfer, episode initialization incorporates targeted randomization. The camera drone starts centered in the arena with gentle perturbations to non-position states (orientation $\times 0.25$, velocity $\times 0.2$, angular rates $\times 0.2$), while the actor follows a RandomTrajectory within 80% world bounds. Critically, the initial orientation aligns the camera such that the actor appears at a random normalized pixel location (uniformly sampled in $[0, 1]^2$), achieved by unprojecting the random pixel to a direction vector and rotating the quaternion to match the true actor direction. This ensures the target is always visible in the initial FoV, preventing early detection failures and accelerating learning by providing consistent starting observations. The bounding boxes are simulated via a double-sphere camera model: 8 corners of a cubic actor model (26 cm diagonal body, 8 cm height) are transformed to world coordinates, projected, clipped to FOV bounds $[-1, 1]$, and used to compute $[x_{\min}, y_{\min}, w, h]$. Invalid projections (degenerate or fully out-of-view) yield sentinel values (-2), emulating real detector outputs under occlusion or distance. This projection-based simulation introduces perspective effects and partial observability, highlighting sim-to-real gaps like unmodeled distortions or noise, while fostering policies robust to deployment variability.

Action Space: Identical to the state-based controller, ensuring consistency in control parameterization.

Neural Network Architecture: Same as the state-based controller, with the MLP extractor accommodating the expanded input dimensionality. This reuse preserves training efficiency while handling temporal visual features.

Reward Function: The vision-based reward extends the state-based structure, adapting to visual proxies and incorporating failure modes for robustness. Shared terms are reused where applicable, with modifications to address partial observability.

$$r_{xy} = \exp\left(-\frac{(d_{xy} - 1.5)^2}{2}\right) + 0.5, \tag{7}$$

identical to (1). This design maintains consistency with the state-based goal of ideal separation.

$$r_z = \exp(-5 \cdot d_z^2), \tag{8}$$

similar to (2) but with a steeper coefficient (-5 versus -1) to amplify sensitivity to altitude errors derived from bounding box depth, critical for FoV stability under noisy measurements.

$$r_{\text{ori}} = \exp(-10 \cdot c_x^4), \tag{9}$$

A term penalizing horizontal bounding box center offset c_x from the image center, absent in the state-based case (where $r_{\text{ori}} = 0$). The quartic form emphasizes precise actor framing, essential for visual servoing, unlike the state-based reliance on direct orientation.

$$r_{\text{vel}} = -0.05 \cdot \exp(-0.3 \cdot d_{3D}) \cdot \|\mathbf{v}_{\text{err}}\|^2, \tag{10}$$

adapting (3) by using actor-relative velocity error $\mathbf{v}_{\text{err}}$ and a lighter gating (-0.3 versus -1.0), reflecting the need for agile responses to noisier position estimates.

$$r_{\text{ang}} = -0.01 \cdot \|\boldsymbol{\omega}_d\|^2, \tag{11}$$

a global penalty (versus gated in (4)) with a reduced coefficient (0.01 versus 0.05) to avoid over-damping during distant pursuits, accommodating vision-based uncertainty.

$$r_{\text{cmd}} = -0.01\left(|\Delta\boldsymbol{\omega}_d| + \|\boldsymbol{\omega}_d\| + |\Delta T| + |T - 1.88|\right), \tag{12}$$

extending (5) to include command deltas ($\Delta\boldsymbol{\omega}_d, \Delta T$), enhancing smoothness under noisy observations while maintaining hover thrust regularization ($T_0 = 1.88$ m/s^2).

$$r_{\text{post}} = 0.2 \exp\left(-|\dot{\mathbf{x}}_b \cdot \mathbf{e}_z|^2\right), \tag{13}$$

a posture reward absent in the state-based case, encouraging forward-facing alignment (body-x velocity $\dot{\mathbf{x}}_b$ with world z-axis $\mathbf{e}_z$), complementing (9) for robust orientation.

The total reward is

$$r = r_{xy} + r_z + r_{\text{ori}} + r_{\text{vel}} + r_{\text{ang}} + r_{\text{cmd}} + r_{\text{post}} + r_{\text{lost}}, \tag{14}$$

with unit weights and $r_{\text{lost}} = -5$ for detection failures. The additive structure preserves interpretability, with (7), (10), (11), (12), and (13) tailored to address vision-specific challenges like noisy detections and FoV retention. The steeper r_z (8) and new terms r_{ori}, r_{post} prioritize visual alignment, while reduced gating in r_{vel}, r_{ang} accommodates uncertainty, balancing agility and stability for deployable vision-based tracking.

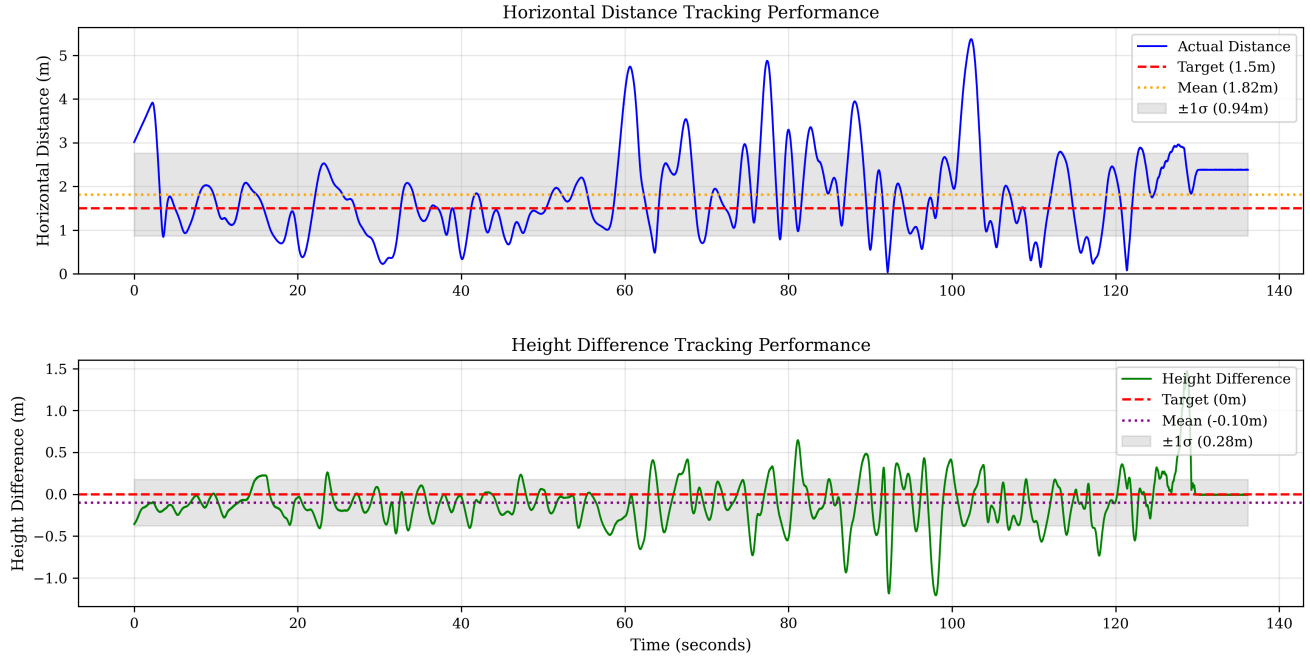

Fig. 1. Real-world tracking performance during autonomous flight testing (136s duration). Top: horizontal distance between camera and actor drones (target: 1.5m). Bottom: height difference tracking (target: 0m). The performance shows the early-stage learning capability of the 800-iteration trained policy under actual flight conditions.

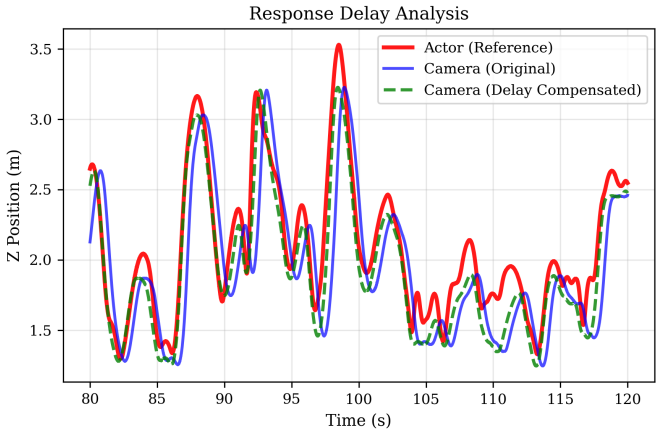

Fig. 2. Response delay analysis for a representative segment (80-120s). Cross-correlation analysis of the Z-axis position data from the full flight (136s) revealed a 0.48s response delay. The delay-compensated trajectory (green dashed) shows strong alignment with the actor reference trajectory (red).

TABLE I
REAL-WORLD TRACKING ERROR STATISTICS

Error Metric	Mean (m)	Std. Dev. (m)
Horizontal Distance	1.82 (1.5)	0.94
Height Distance	-0.10 (0.0)	0.28

III. EXPERIMENTAL RESULTS

Table I shows the controller's tracking performance during the real-world flight test. The controller maintained a mean horizontal stand-off distance of 1.82 m, which is close to the target of 1.5 m, with a standard deviation of 0.94 m. The performance in altitude tracking was highly accurate, showing a minimal height of -0.10 m (targeting 0.00 m) and a low standard deviation of 0.28 m.

We successfully deployed our state-based controller in the real world and recorded the rosbag during flight for evaluation. The control objective is to keep a 1.5 m stand-off in the horizontal plane and to minimize the altitude mismatch to the actor. We used a policy trained for 800 iterations with default PPO parameters and other training configurations. Fig.1 presents the tracking performance analysis over the complete flight duration, while Fig.4 provides response delay analysis using cross-correlation of Z-axis position data. These metrics directly correspond to the tracking and smoothness terms used during training. The policy demonstrates stable tracking performance, successfully maintaining autonomous flight throughout the entire duration despite limited domain randomization during training, indicating effective sim-to-real transfer capabilities for this tracking task. The controller consistently tracks the actor drone across various flight scenarios, demonstrating the robustness of the learned behaviors. However, there remains potential for further optimization in tracking precision and response characteristics. When encountering larger deviations from the target distance, the convergence rate could be enhanced for more responsive tracking. This behavior is likely attributed to the limited training iterations (800), where the policy may not have fully explored recovery strategies for large tracking errors. Extended training could potentially improve convergence speed and overall tracking performance.

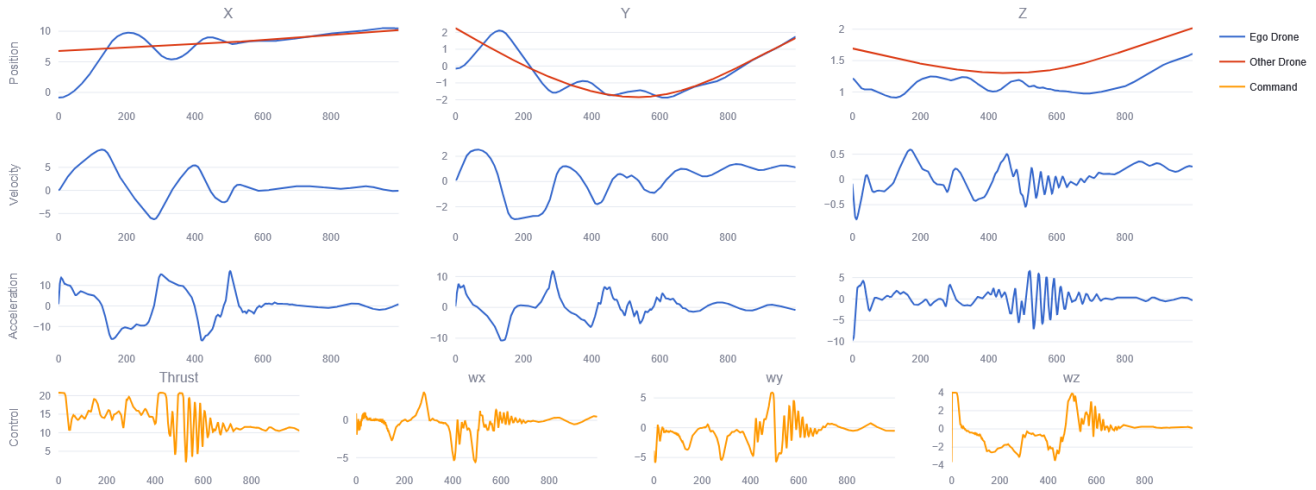

Fig. 3. Simulation plots of vision based drone

As we could not deploy the vision-based policy, our results derive from the simulation. Fig. 3 shows the simulation plots of our vision-based policy. While the drone tracks the target well, the extreme variation of the control signals indicates instability, which could be dangerous if deployed. This could be due to the control inputs not being penalized enough, leading to the drone over-correcting as it gets closer to the target drone. This was exacerbated by the high reward given to tracking, which could have led the drone to prioritize closeness to the target rather than self-stability. In other tested reward functions, this behavior also manifested as the drone crashing seemingly deliberately, as a crash was considered to give more reward than any possible action. Another item of note is that the while the state-based policy excelled in maintaining the same height as the actor drone, the vision-based policies were particularly deficient in this - while the xy-tracking performed well, the z-tracking was always about 1 meter off-target. This behavior persisted across a variety of reward functions and random simulations.

IV. Discussion

To address the challenges of vision-based drone tracking, we explored an alternative reward function inspired by the dense, shaped reward formulation proposed in [2]. This approach, designed for perception-aware autonomous drone racing, combines progress, perception, action smoothness, and collision penalties to guide policy learning. We adapted this framework to our state-based controller, aiming to enhance tracking performance by emphasizing stable flight and precise target following. The implemented reward function is defined as:

$$r = r_{\text{track}} + r_{\text{ori}} + r_{\text{vel}} + r_{\text{ang}} + r_{\text{act}}, \tag{15}$$

where:

- $r_{\text{track}} = \frac{1}{1+\|\mathbf{p}_d - \mathbf{p}_a\|}$ rewards proximity to the actor, providing a smooth gradient for convergence, similar to the progress term $r_{\text{prog}} = \lambda_1[d_{t-1}^{\text{Gate}} - d_t^{\text{Gate}}]$ in [2], but using an inverse distance to prioritize absolute closeness over incremental progress.
- $r_{\text{ori}} = 0.1 \cdot \exp(-\|\mathbf{p}_d - \mathbf{p}_a\|) \cdot (\mathbf{R}_d[:,2] \cdot \mathbf{e}_z)$ encourages upright orientation, gated by distance to focus on close-range stability, contrasting with the perception term $r_{\text{perc}} = \exp[-\lambda_2\lambda_3\delta_{\text{cam}}^4]$ in [2], which aligns the camera's optical axis with the target gate.
- $r_{\text{vel}} = -0.1 \cdot \exp(-\|\mathbf{p}_d - \mathbf{p}_a\|) \cdot \|\mathbf{v}_d\|^2$ penalizes excessive linear velocity to promote hovering, akin to the action smoothness term $r_{\text{cmd}} = \lambda_4\|\mathbf{a}_t\|^2 + \lambda_5\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$ in [2], but gated to emphasize stability near the actor.
- $r_{\text{ang}} = -0.05 \cdot \exp(-\|\mathbf{p}_d - \mathbf{p}_a\|) \cdot \|\boldsymbol{\omega}_d\|^2$ curbs rapid rotations, further refining smoothness.
- $r_{\text{act}} = -0.001 \cdot \exp(-\|\mathbf{p}_d - \mathbf{p}_a\|) \cdot \left(\|\pi_{\text{act}}[1:4]\|^2\right)$ regularizes control inputs, mirroring r_{cmd}.

This reward structure excelled in drone racing [2], leveraging predictable gate positions to align the camera via r_{perc} and ensure agile navigation with r_{prog} and r_{cmd}. However, when applied to our vision-based controller, it failed to mitigate instabilities in bounding box-based tracking, such as oscillatory pursuits and FoV losses. The racing task benefits from low-noise visual cues and static targets, but our dynamic actor introduces scale ambiguity and stochastic noise in bounding boxes. The r_{ori} term, focused on upright posture, does not address precise FoV centering, critical for visual servoing. The absence of a target-lost penalty (unlike $r_{\text{lost}} = -5$ in (14)) fails to handle frequent detection failures from occlusions, and velocity penalties neglect relative errors ($\mathbf{v}_{\text{err}}$) needed for dynamic tracking, unlike (10). The racing environment's structured priors contrast with our randomized trajectories,

exacerbating partial observability challenges.

Additionally, inspired by [2], we simplified the vision-based observation space to an 18D vector, comprising the drone's state (3D position $\mathbf{p}_d$, 6D rotation $\mathbf{R}_d$, 3D velocity $\mathbf{v}_d$, 3D angular velocity $\boldsymbol{\omega}_d$) and only the current 3D actor position $\mathbf{p}_a$, omitting temporal history (previous and two-steps-prior positions or bounding boxes). This reduction, intended to streamline learning by mimicking the racing task's simpler state inputs, failed to provide sufficient context for motion prediction under noisy bounding box inputs. Unlike the 27D observation space in our primary vision-based design, which includes temporal bounding box sequences to mitigate noise and occlusions, the simplified 18D space lacks the predictive capacity to handle dynamic actor motion and perceptual aliasing, leading to unstable tracking and frequent FoV losses. The racing task's static gates allow a compact state representation, but our dynamic, noisy environment demands richer temporal data for robust inference.

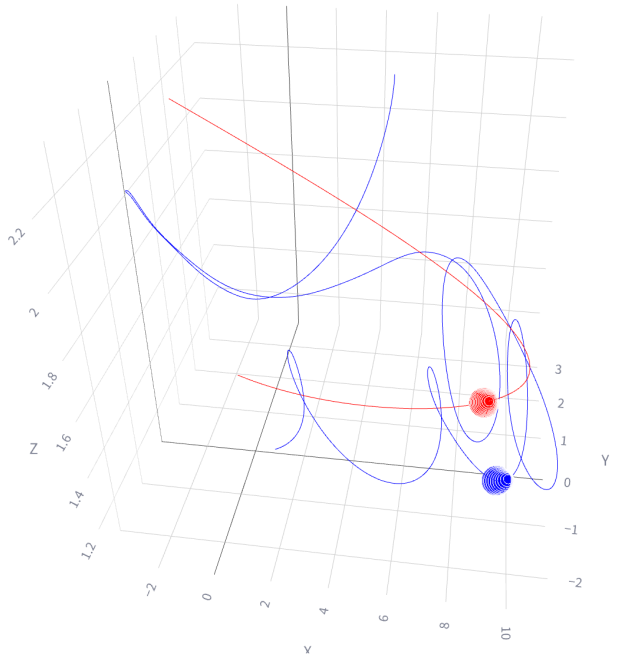

Fig. 4. Trajectories from a simulation experiment illustrating the undesired orbiting behavior of the policy-controlled drone (blue) around the target drone (red). This phenomenon arises when the follower drone overspeeds while attempting to maintain visual contact with the target.

During our simulation experiments, we observed an orbit-like behavior of the policy-controlled drone around the target drone. We hypothesize that this phenomenon arises because the policy drone is overspeeding while simultaneously attempting to maintain visual contact with the target drone, particularly when orientation-related perception errors occur. In such cases, the most feasible strategy for the policy drone is to orbit around the target rather than follow it directly from behind. To mitigate this issue, we introduced a velocity penalty modulated by a distance-dependent gate: when the policy drone is sufficiently close to the target drone, it is encouraged to match the target's velocity. This modification partially alleviated the undesired orbiting behavior in the simulation. However, this approach is difficult to deploy in real-world scenarios, as the target drone's velocity is typically unavailable and cannot be reliably estimated from a single bounding box observation alone. Future work may require developing more robust velocity inference techniques—such as using temporal information from multiple frames—or designing policies that rely less on explicit velocity matching while still ensuring stable following behavior.

V. Conclusion

We successfully developed and validated a state-based RL controller for vision-guided drone tracking, achieving precise FoV maintenance in both simulation and deployment. The vision-based extension, while unsuccessful in converging, provided valuable insights into perception-control integration challenges. This project demonstrates RL's promise for autonomous flight while highlighting the need for robust observation models.

References

[1] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[2] Kaufmann, Elia, et al. "Champion-level drone racing using deep reinforcement learning." Nature 620.7976 (2023): 982-987.