Vision Based Drone Flight

Rajiv Bharadwaj
D-MAVT
ETH Zurich
rbharadwaj@ethz.ch

Riccardo Bianco
D-MAVT
ETH Zurich
rbianco@ethz.ch

Lorenzo Rocco Didò
D-MAVT
ETH Zurich
lodido@ethz.ch

Roham Zendehdel Nobari
IFI UZH
University of Zurich
roham.zendehdelnobari@uzh.ch

Nathan Lacour
D-MAVT
ETH Zurich
nlacour@ethz.ch

***Abstract*—Autonomous, vision-based drone flight enables applications in inspection, mapping, and search-and-rescue, where a vehicle must perceive and react to dynamic environments using only onboard cameras. In this project, we address vision-based pursuit: controlling a "camera drone" to keep an "actor drone" within the camera's field of view at a target distance. We focus on the control component (Task 2) and assume a bounding-box detector provides the actor's image-space location at 60 Hz. Using Flightmare and Agilicious, we design a PPO-based reinforcement learning controller trained in simulation with dynamics randomization, and deploy it via ROS at 60 Hz. With ground-truth bounding boxes, the policy reliably tracks the actor in both simulation and real-world tests, performing stable pursuit maneuvers. However, integrating the controller with a partner team's real-time detector led to rapid performance degradation and loss of stability, preventing a successful full closed-loop demonstration. [Video]**

I. Introduction

Vision-based autonomy is a key enabler for drones operating in inspection, mapping, and search-and-rescue, where vehicles must perceive and react in real time using only onboard cameras. In this project, we study vision-based pursuit: controlling a "camera drone" to keep an "actor drone" inside the field of view at a target distance, using camera images to estimate the actor drone.

The control task is formulated as a reinforcement learning problem: the camera drone must maintain distance to the actor while keeping it visible, all under smooth and stable flight dynamics. Proprioceptive measurements are combined with image-space bounding box estimates of the actor to output thrust and body rate commands.

We use an incremental approach, starting from hovering and tracking using full actor state information to estimating the ground truth bounding box using a double sphere camera projection model [1] before we experiment with real time bounding box deployments. We use Proximal Policy Optimization (PPO) [2] with domain randomization to learn robust tracking policies.

Our experiments demonstrate robust real-world tracking of the actor drone when ground-truth bounding boxes are available, while also meeting secondary objectives. However, performance degrades with real-time detection due to modeling mismatches and real-world uncertainties. This report summarizes the methodology, results, and recommendations for future end-to-end vision-based tracking approaches.

II. Methodology

A. Rewards:

1) Hovering task: : This task aimed to make the drone hover from every point in the world bounding box to the point $[0, 0, 1]^T$. To achieve this, we formulate the reward function on the position to be a Laplacian distribution (Fig.1a) to enforce the camera drone to be really close to the target position. The position reward is the main component of the total reward, while the action reward (trust and body rates) avoids oscillations around the target point.

Let $\mathbf{p} \in \mathbb{R}^3$ be the quadrotor position, $\mathbf{u} = \begin{bmatrix} u_T & \boldsymbol{\omega} \end{bmatrix}^\top$ the action composed of thrust $u_T \in \mathbb{R}$ and body rates $\boldsymbol{\omega} \in \mathbb{R}^3$. The hover targets are $\mathbf{p}_{\text{ref}} = [0,\ 0,\ 1]^\top$, $u_{T,\text{ref}} = 9.81$, $\boldsymbol{\omega}_{\text{ref}} = \mathbf{0}$.

a) Position tracking:

$$r_{\text{pos}} = \alpha_p \exp\big(- \|\mathbf{p} - \mathbf{p}_{\text{ref}}\|_1 \big) \tag{1}$$

b) Action tracking:

$$r_T = \alpha_T \big\| u_T - u_{T,\text{ref}} \big\|_2 \tag{2}$$

$$r_\omega = \alpha_\omega \big\| \boldsymbol{\omega} - \boldsymbol{\omega}_{\text{ref}} \big\|_2 \tag{3}$$

c) Total reward:

$$r = r_{\text{pos}} + r_T + r_\omega + \alpha_{\text{col}}\, \phi_{\text{col}} \tag{4}$$

where $\phi_{\text{col}} \leq 0$ is a (task-dependent) collision penalty signal. All the coefficients α are listed in the Tab.II.

2) Tracking state trajectory task: We augment the reward design to track the state-based trajectory of the actor drone in place of a fixed hovering point at a distance of 1.5 m.

Let $\mathbf{p}, \mathbf{p}^a \in \mathbb{R}^3$ denote the camera and the actor drone positions.

Unlike hovering, we design asymmetrical position tracking rewards, with higher penalties as the actor drone gets too close in order to enforce a safe following distance. This is achieved by steepening the inner side of the reward (Fig.1b). We also decouple the tracking in the horizontal plane (x and y) from the vertical direction (z), since maintaining $|\Delta_z| = |z - z_a| \approx 0$ reduces control complexity and supports the subsequent visual tracking task by ensuring the drone remains visible to the camera.

$$d_{xy} = \|\Delta_{1:2}\|_1, \qquad d_z = |\Delta_z|.$$

a) Position reward:

$$h(x; d_{\text{des}}) = \begin{cases} \exp\big(-3\,|\,|x| - d_{\text{des}}\,|\big) & |x| < d_{\text{des}} \\ \exp\big(-|\,|x| - d_{\text{des}}\,|\big) & |x| \geq d_{\text{des}} \end{cases} \tag{5}$$

$$r_{\text{pos},xy} = \alpha_{p,xy}\,\frac{2}{3}\,h(d_{xy}; d_{\text{des}}) \tag{6}$$

$$r_{\text{pos},z} = \alpha_{p,z}\,\frac{1}{3}\,\exp\Big(-\frac{d_z^2}{2\cdot 0.3^2}\Big) \tag{7}$$

b) Total reward.:

$$r = r_{\text{pos},xy} + r_{\text{pos},z} + r_T + r_\omega + \alpha_{\text{col}}\,\phi_{\text{col}} \tag{8}$$

The policy trained using 6 experienced crashes due to insufficient penalization. We switch to a linear negative reward if $d_{xy} < 1.5m$ and a wide gaussian if $d_{xy} \geq 1.5m$ for increasing the area of close to optimal reward outside the constraint of $d = 1.5m$ (Fig.1c), yielding safer tracking performance.

$$h'_n(x; d_{\text{des}}) = \begin{cases} -3\,|\,|x| - d_{\text{des}}\,| + 1, & |x| < d_{\text{des}}, \\ \exp\Big(-\frac{\left(|\,|x| - d_{\text{des}}\,|\right)^2}{2\cdot 0.4^2}\Big), & |x| \geq d_{\text{des}} \end{cases} \tag{9}$$

$$r'_{\text{pos},xy} = \alpha_{p,xy}\,\frac{2}{3}\,h'(d_{xy}; d_{\text{des}}) \tag{10}$$

c) Total reward.:

$$r = r'_{\text{pos},xy} + r_{\text{pos},z} + r_T + r_\omega + \alpha_{\text{col}}\,\phi_{\text{col}} \tag{11}$$

3) Ground Truth Bounding Box Task: We add additional reward terms to describe the expected bounding box behavior, while keeping the tracking rewards largely the same from before so that the policy is able to learn the correlations in state and bounding box dynamics. Since this task is harder than the previous tasks, we substitute the linear reward on the xy position if $d < 1.5m$ with a Gaussian penalty Eq:13 (Fig.1d) to prevent jittery actions.

Define the bounding box represented in pixel coordinates as

$$\mathbf{b} := \begin{bmatrix} x_{\min} & y_{\min} & x_{\max} & y_{\max} \end{bmatrix} \tag{12}$$

For an image of width W and height H, the normalized bounding box center is

$$\mathbf{c}_{\text{bb}} = \begin{bmatrix} \frac{x_{\min} + x_{\max}}{2W} & \frac{y_{\min} + y_{\max}}{2H} \end{bmatrix}^\top, \mathbf{c}_{\text{img}} = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}^\top$$

$$d_{\text{center}} = \|\mathbf{c}_{\text{bb}} - \mathbf{c}_{\text{img}}\|_1$$

a) Position reward:

$$h''(x; d_{\text{des}}) = \begin{cases} 8\,\exp\big(-\frac{1}{2}\big(|\,|x| - d_{\text{des}}\,|\big)^2\big) - 7 & |x| < d_{\text{des}}, \\ \exp\big(-0.4\,(|\,|x| - d_{\text{des}}\,|)^2\big) & |x| \geq d_{\text{des}} \end{cases} \tag{13}$$

$$r''_{\text{pos},xy} = \alpha_{p,xy}\,\frac{2}{3}\,h''(d_{xy}; d_{\text{des}}) \tag{14}$$

b) Bounding box reward:

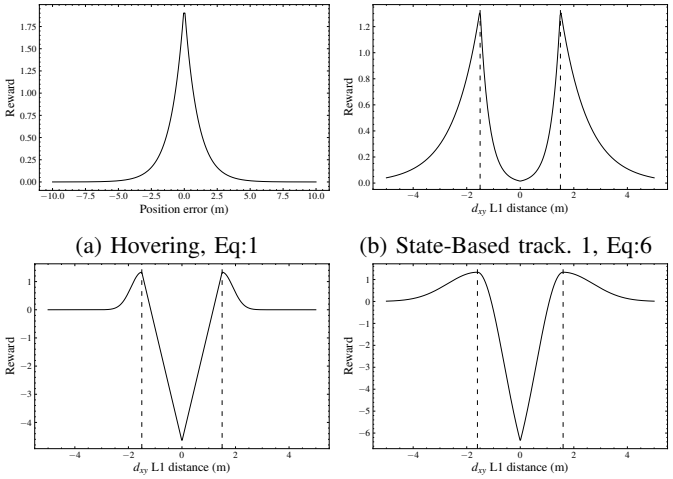
$$r_{\text{bb}} = \alpha_{\text{bb}}\Big(6\,\exp(-2\,d_{\text{center}}^2) - 5\Big)\cdot \mathbb{I}\{\text{bbox detected}\} \tag{15}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function.

c) Total reward:

$$r = r''_{\text{pos},xy} + r_{\text{pos},z} + r_{\text{bb}} + r_T + r_\omega + \alpha_{\text{col}}\,\phi_{\text{col}} \tag{16}$$

When switching to real Bounding Boxes, we have maintained the same reward function.



(a) Hovering, Eq:1 (b) State-Based track. 1, Eq:6

(c) State-Based track. 2, Eq:10 (d) Bounding Box track. Eq:14

Fig. 1: Rewards functions development: - - - desired position, — reward function

B. Observation Space

Across all tasks, the observation space includes the ego quadrotor state and the previous action, with task-specific extensions summarized in Tab. I and described below:

- For the hovering task, we do not utilize additional terms.
- For the position tracking task, two formulations are considered. The first (Eq. 6) also includes the ground-truth actor position to define the target to be tracked. The second (Eq. 10) uses a short buffer of past observations is added to provide temporal context, enabling the policy to infer latent dynamics such as the relative velocity of the drones.
- For the visual tracking task, we replace the actor position with the detected bounding box coordinates of the actor drone. We continue using the buffer to store past bounding box observations (Sec. II-A3). This allows the policy to infer the velocity of the target from the motion of the bounding box center relative to the image center, enabling it to anticipate fast movements and follow the actor.
- For real time bounding box observations, we also provide a running average representation of recent frames to improve robustness against real world noise and detection dropouts.

C. Actor drone trajectories for training

To train a robust tracking policy we employ three classes of actor drone trajectories (2), each offering various trade-offs between smoothness, realism, and generalization.

TABLE I: Unified observation spaces and feature descriptions

(a) Observation space across tasks

Task	Ego pos.	Ego rot.	Ego lin. vel.	Ego ang. vel.	Prev. act.	Actor pos.	Actor bbox	Stack. obs.	Avg. obs.
Hovering	✓	✓	✓	✓	✓	–	–	–	–
Trajectory tracking (1, 2)	✓	✓	✓	✓	✓	✓	–	✓	–
Visual tracking (sim. bbox)	✓	✓	✓	✓	✓	–	✓	✓	–
Visual tracking (real bbox)	✓	✓	✓	✓	✓	–	✓(noisy)	✓	✓

(b) Feature dimensions and descriptions

Feature	**Dimension**	**Description**
Ego position	$\mathbb{R}^3$	Cartesian position $\mathbf{p}$
Ego rotation	$SO(3)$ (flattened)	Rotation matrix $\mathbf{R}$
Ego linear velocity	$\mathbb{R}^3$	Linear velocity $\mathbf{v}$
Ego angular velocity	$\mathbb{R}^3$	Angular velocity $\boldsymbol{\omega}$
Previous action	$\mathbb{R}^4$	Action at $t-1$, $\mathbf{u}_{t-1}$
Actor position	$\mathbb{R}^3$	Ground-truth position of actor drone $\mathbf{p}^a$
Actor bounding box	$\mathbb{R}^4$	Normalized $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$
Noisy bounding box	$\mathbb{R}^4$	Bounding box with added measurement noise
Stacked observations	$3\times$ above	Buffer of 2 past + current frames
Averaged observation	same as above	Element-wise mean of the last 3 stacked observations

1) Random Hermite Trajectory: Generated by cubic Hermite splines through random waypoints. Each segment is of the form

$$\mathbf{p}(u) = \mathbf{a} + \mathbf{b}u + \mathbf{c}u^2 + \mathbf{d}u^3, \quad u \in [0,1],$$

with tangents chosen as forward-facing bisectors to ensure smoothness. Speed is modulated by curvature, with slowdowns in sharp turns, and a boundary-repulsion keeps the trajectory inside the world bounds.

2) Lissajous Trajectory: Defined analytically as

$$\mathbf{p}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} c_x + A_x \sin(\omega_x t + \phi_x) \\ c_y + A_y \sin(\omega_y t + \phi_y) \\ c_z + A_z \sin(\omega_z t + \phi_z) \end{bmatrix},$$

where (c_x, c_y, c_z) is the center, A_i the amplitude, ω_i the frequency, and ϕ_i the phase. These trajectories are smooth, continuous, and repeatable, but their regularity can lead to overfitting.

3) Waypoint-Based Random Trajectory: Constructed by sampling N waypoints $\{\mathbf{p}_0, \ldots, \mathbf{p}_N\}$ within the simulated world and connecting them with straight segments:

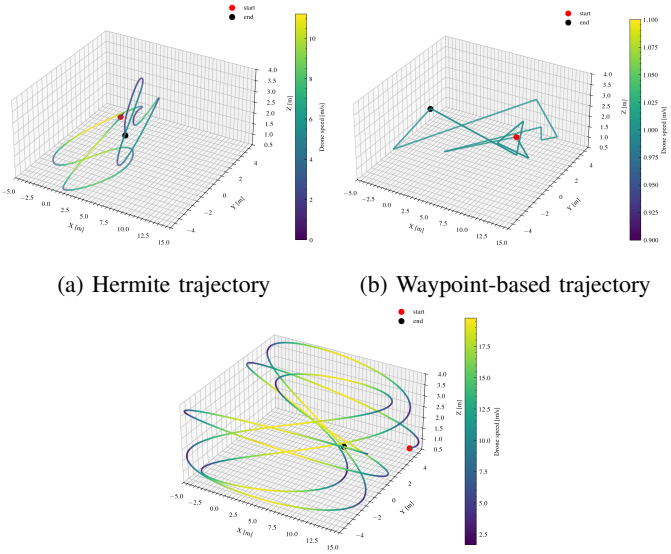
$$\mathbf{p}(t) = \mathbf{p}_i + \alpha(t)\,(\mathbf{p}_{i+1} - \mathbf{p}_i),$$

where $\alpha(t) \in [0,1]$ is chosen such that the drone moves at constant speed along each segment. This produces discontinuous derivatives at waypoints, capturing abrupt maneuvers typical of real drone flight.

D. Domain Randomization

To ensure robustness against sim-to-real uncertainties, we train the policy against random initial states of the ego drone and also vary the dynamics between episodes.

For tracking tasks, it is important to ensure that the actor drone covers the entire workspace to prevent overfitting. In state based tracking, we randomly initialize actor in the workspace and apply randomizations to the curve parametrization in (Sec. II-C) for diverse behaviors.



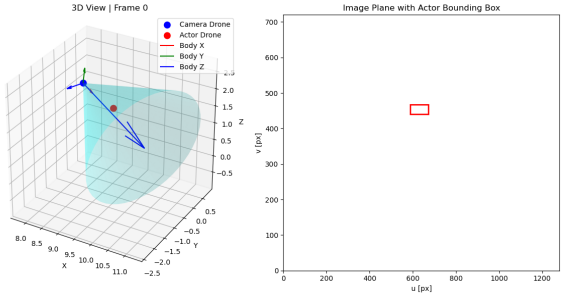
(a) Hermite trajectory (b) Waypoint-based trajectory

(c) Lissajous trajectory

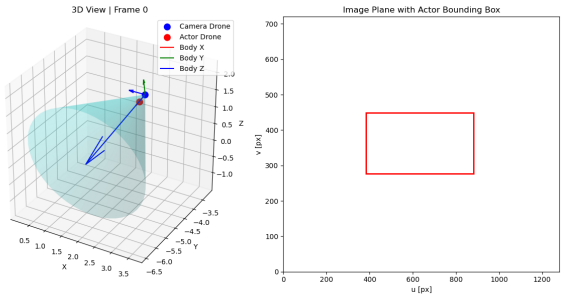
Fig. 2: Examples of reference trajectories used for training and evaluation of the vision-based drone controller. All trajectories are generated within the predefined world bounds.

For visual tracking, we extend the initialization of the camera drone to ensure the actor drone is always visible in the field of view of the camera, but never in the exact center of the image.

a) Camera Drone Field of View Initialization: At the beginning of each episode, the actor drone is placed according to its randomly generated trajectory. The camera drone is then initialized on a sphere of radius $R = 1.5\,\text{m}$ around the actor,



(a) Episode 1 Initialization



(b) Episode 2 Initialization

Fig. 3: Initialization of actor and camera drones in two training episodes. In (b), the camera drone's initializing sphere radius R is shrunk to keep the drone within the world bounds. The big blue arrow shows the camera drone's optical axis.

with a random yaw offset $\psi \sim \mathcal{U}[-\pi, \pi]$, pitch offset $\phi \sim \mathcal{U}[-10°, 10°]$, and a small vertical jitter $\delta z \sim \mathcal{N}(0, 0.15^2)$. The resulting direction vector from actor to camera is

$$\mathbf{u}_{ac} = \begin{bmatrix} \cos(\phi)\cos(\psi) \\ \cos(\phi)\sin(\psi) \\ \sin(\phi) \end{bmatrix},$$

and the suggested camera position is

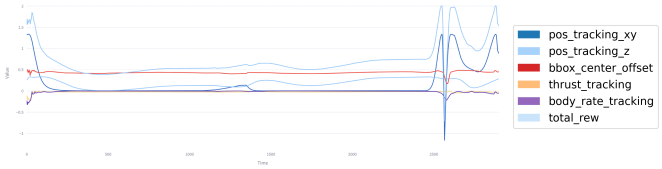
$$\mathbf{p}_c = \mathbf{p}_a + R\,\mathbf{u}_{ac} + [0, 0, \delta z]^\top,$$

where $\mathbf{p}_a$ is the actor position. If $\mathbf{p}_c$ lies outside the world boundaries, the radius R is shrunk to remain within a margin of the bounds. The optical axis of the camera is aligned with the actor by defining

$$\mathbf{f} = \frac{\mathbf{p}_a - \mathbf{p}_c}{\|\mathbf{p}_a - \mathbf{p}_c\|},$$

and constructing an orthonormal frame $(\mathbf{x}_W, \mathbf{y}_W, \mathbf{f})$ with $\mathbf{y}_W = \mathbf{f} \times \mathbf{x}_W$. The world-to-camera rotation R_{WC} is then transformed into the drone body frame using the extrinsic calibration R_{BC} of the onboard camera. Finally, a random roll perturbation $\theta \sim \mathcal{U}[-15°, 15°]$ is applied around the optical axis to avoid having the actor drone in the center of the image.

b) Real bounding boxes: When moving from simulated to real detections, the bounding box of the actor drone is no longer computed from ground-truth position, but provided by a real-time detector. Such detections are inherently noisy due to measurement errors, illumination changes, and missed



(a) 600th iteration

(b) 900th iteration

Fig. 4: Reward components during training in the visual tracking task. After 600 iterations (a), the policy maximizes the bounding box centering reward while the position reward remains near zero; by 900 iterations (b), the position reward starts improving while the centering reward stays high.

frames. To increase robustness, we enhance our training setup by injecting additive pixel noise into the simulated bounding boxes from equation 12:

$$\tilde{\mathbf{b}} = \mathbf{b} + \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{U}(-5, 5),$$

This forces the policy to learn robustness against small, random shifts in the bounding box coordinates.

III. Training and Evaluation

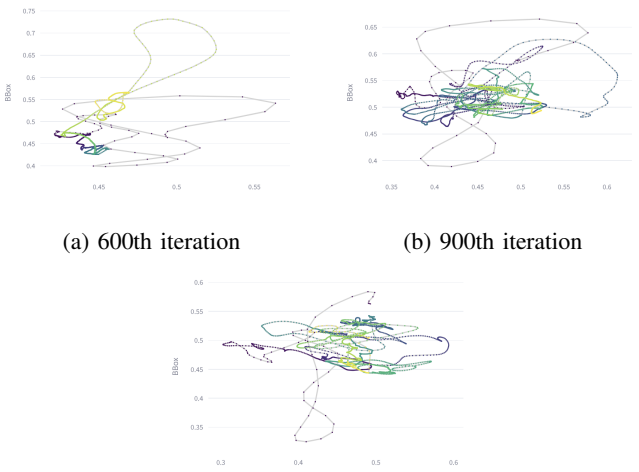
We use Proximal Policy Optimization [2] to train the policy for each of the tasks over the defined reward scheme, observation, and action spaces. By evaluating the policy performance against an evaluation environment every 100 iterations, we are able to view metrics on the estimated accumulated reward as in Fig.4, or the bounding box tracking performance seen in Fig.5 to determine whether the learning has stabilized.

TABLE II: Reward function coefficients for the different tasks.

Task	α_p	$\alpha_{p,xy}$	$\alpha_{p,z}$	α_{bb}	α_T	α_ω	α_{col}
Hovering	2.0	–	–	–	-0.02	-0.04	-16.0
Traje. Track. 1	–	2.0	2.0	–	-0.02	-0.04	-16.0
Traje. Track. 2	–	2.0	2.0	–	-0.02	-0.04	-16.0
Simu. BBox	–	2.0	1.0	0.5	-0.02	-0.04	-16.0

A. Iterative Reward Tuning using Evaluations

Across all tasks the reward structure follows a common scheme. Cost terms on thrust and body rates are kept identical, while the position-related component is adapted for each task. This can be seen from the equations in Sec.II-A and the coefficients Tab.II. As we add more reward terms, it is important to control the relative importance between each term to ensure the policy learns realistic behavior. We utilize the evaluation episodes to help guide this design.



(a) 600th iteration (b) 900th iteration

(c) 4000th iteration

Fig. 5: Bounding box tracking during training. The plots show the trajectory of the bounding box center during different training iterations. The policy already learns to consistently keep the target within the image after 600 iterations, with no substantial changes observed in later training stages.

For visual tracking, it important to carefully weigh the bounding box reward terms so as to maintain their secondary nature for the task and help guide the policy to learn the implicit dynamics between the bounding box and state information.

As shown in Fig. 4, the policy learns to maximize the bounding box centering reward much earlier than the distance reward. This likely stems from two factors: (i) bounding box centering is directly tied to an observable feature, while distance must be inferred indirectly from the bounding box geometry; and (ii) due to quadrotor dynamics, keeping the target centered can be achieved at large distances with small yaw adjustments, making visual centering easier when the drones are far apart. Over-weighting this term would therefore bias the policy toward staying too far from the target, rather than learning a balanced trade-off between visual centering and distance regulation.

This design allows the policy to associate bounding box geometry with position based rewards to maintain a distance of 1.5m from the actor

IV. Experimental Results:

A. Setup

1) Simulation Environment: We use the Agilicious simulator [3] in ROS to evaluate the camera drone's policy. The simulator provides accurate robot states and dynamics, and allows us to send commands to the actor drone to test how the camera drone responds. We use the double-sphere camera model [1] to estimate the bounding box around the actor drone's state when deploying the visual bounding box tracking task II-A3

During each episode, we record drone states, odometry, and policy actions, which are used to analyze trajectory smoothness, tracking performance, and hovering stability.

2) Real World Deployment: Experiments were conducted on the physical camera and actor drones using ROS. Accurate state and odometry information was provided by means of on-board sensors and motion capture cameras covering the deployment area. The double-sphere camera model [1] was applied on the estimated actor drone positions, retrieved using motion capture, to deploy the visual bounding box tracking task II-A3.

B. Results

Our policy achieves the goal of each task: hovering, tracking, and vision based tracking in a controlled and stable fashion when provided with actor state information from the motion capture system. We discuss the performance against real-time vision based tracking in the next section.

1) State-based tracking: By designing a reward function with a steep drop-off from the desired holding point, the robot learns to stabilize quickly and efficiently to the holding point, as seen in Figure 6. The penalization of input actions encourages the drone to stabilize quickly without any rotational moments as seen from the body rates

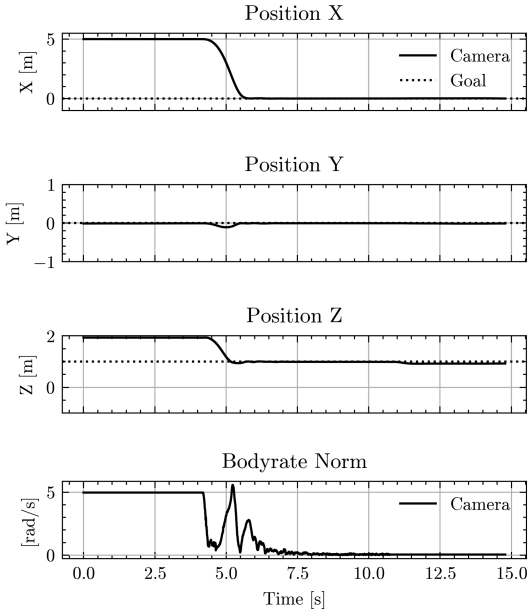


Fig. 6: Time series Position and Body Rate for Hovering

2) Vision Based Tracking: The drone is able to achieve the tracking trajectory task with stable and controlled behavior while staying close to the desired distance of 1.5m in the XY plane and maintaining the same height in the Z axis as seen in Figure 7. Table III shows the average tracking error for each of the tasks during the real world deployment.

Discussion

While the Lissajous trajectory (Sec. II-C2) provides a smooth curve of states for the policy to learn to follow, real

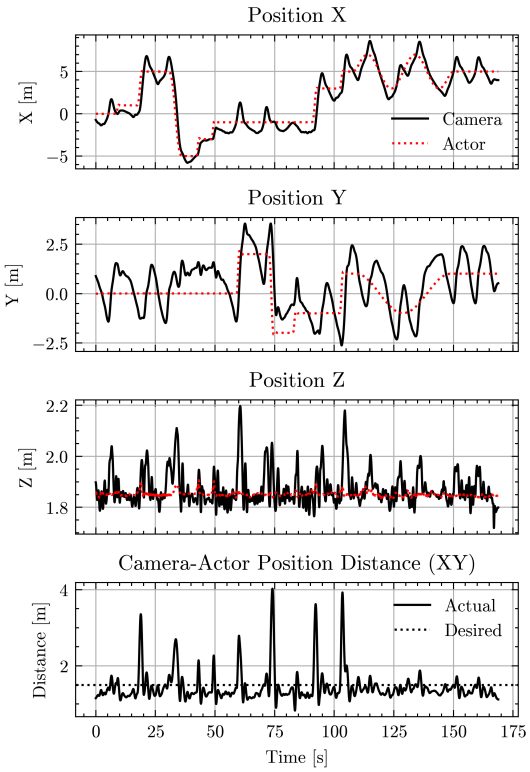


Fig. 7: Time series state and input data for simulated bounding box tracking

TABLE III: Average Tracking Error per Task

Task	Hovering	St. Tracking	Sim. BBox	Real BBox
XY (m)	0.0090	0.0009	0.0380	-
Z (m)	0.0388	0.1385	0.04677	-

world deployments often resulted in crashes since these curves did not represent the agility of the camera drone and its subsequent ability to exert abrupt linear and angular velocity changes. Using random piecewise linear waypoint trajectories (Sec. II-C3) led to improved agility and tracking accuracy. We propose future approaches to extend domain randomization to the actor trajectory types for the policy to experience a more complete representation of the actor drone's behavior.

The policy transferred from simulation to the real world with minimal issues. In addition to a simple formulation of the quadrotor dynamics in simulation and a closed deployment environment without external disturbances; applying domain randomization over the dynamics, and modeling communication delays of 20ms helped improve the robustness of the policy in the real world.

When deploying against real time bounding box information, the drone experienced an immediate failure to maintain stable and controlled flight. Fig.8 shows the difference in normalized coordinates between the real time estimations vs. modeled bounding boxes (computed post deployment using recorded motion capture states). We notice that there is a discrepancy in both the center and the shape of the boxes. This could be attributed to modeling inaccuracies between the double sphere camera model and the real world behavior of the estimator. Since the quadrotor observes a history of bounding boxes, this discrepancy results in the policy experiencing a different dynamical behavior of the bounding boxes than the one learned during training, leading to immediate loss of control and failure.

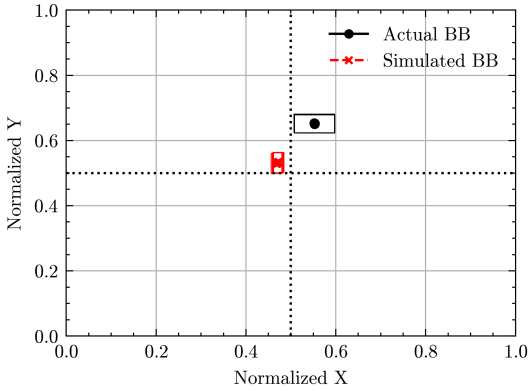


Fig. 8: Difference in estimated vs real bounding box centers

We propose enhancing the domain randomization (in addition to pixel shift) with random rotations, scaling, and affine transforms to improve the robustness against such sim-to-real gaps.

Conclusion

We developed and deployed a reinforcement learning controller for vision-based pursuit, mapping bounding-box observations and drone state to low-level thrust and body-rate commands. The policies we trained achieved stable hovering, state-based tracking, and robust visual tracking when using ground-truth or simulated bounding boxes, with minimal sim-to-real gap for dynamics. However, end-to-end integration with a real-time detector failed to sustain stable closed-loop flight due to mismatch between simulated and real bounding boxes. This identifies the detection-to-control interface as the primary bottleneck. Future work should focus on closing this gap through improved camera/detector modeling in simulation, stronger observation randomization, temporal filtering and history augmentation, and joint sim-in-the-loop tests with the actual detector.

References

[1] V. Usenko, N. Demmel, and D. Cremers, "The double sphere camera model," *CoRR*, vol. abs/1807.08957, 2018. arXiv: 1807.08957.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347.

[3] Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, Davide Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *AAAS Science Robotics*, 2022.