Reinforcement Learning-Based Control of a Quadcopter

Luis A. M. Almeida lmeira2001@gmail.com

Hanwei Zhou

1. Project Objective

The main focus of the project was to investigate some of the existing control methodologies for quadcopters by implementing an RL model, which can perform stable hovering and point-to-point flight. More specifically, this work is focused on the design, training, and testing of a model that would be able to control a simulated quadcopter to reach a certain position in space and remain stable at that position [2].

This work is an exploratory exercise to apply state-of-the-art RL techniques, such as Soft Actor-Critic, to the control of quadcopters. The project focused on stable hovering and position tracking, whereas the potential of the RL models for handling dynamic, stochastic environments, such as wind disturbances and sensor noises, was tested to show the generalization capabilities of such a model [4]. Especially since these are areas typically out of reach of traditional control systems.

Beyond the main task, a few other tests were also conducted to assess the capabilities of the RL model. These included the addition of noise to the original task, trajectory tracking along a straight line, a circular path, and a robustness test simulating extreme initial conditions. These tests were not core objectives but were included to show the potential and adaptability that the RL model can achieve.

The project also provides a practical baseline for performance evaluation of the RL model by implementing and comparing a traditional PID controller [12]. This will, in turn, highlight some of the strengths and limitations of RL-based approaches in light of established control methods.

2. Motivation

The increased application of drones, from delivery services to farming, disaster response, intelligence, surveillance, and reconnaissance, calls for advanced controlling systems operating in widely varying and unpredictable environments [11]. Traditional controllers work well in structured and highly predictable scenarios but fail in areas with stochastic elements, including strong gusts of wind or sensor noise [5]. RL has been shown to work well in these scenarios, having frameworks that can learn optimal control policies by training rather than relying on precise mathematical models.

This project was also motivated by the team's interests and backgrounds in mechanical engineering and the interest in using the latest technologies for solving practical problems. This project provided an opportunity to investigate the fast-growing field of drone control systems, building upon other research projects of one of the team members.

3. Background

First, the dynamics governing flight for quadcopters, second, the principles and equations of PID controllers, and third, the methodology and mathematical formulation of the Soft Actor-Critic algorithm represent the core concepts on which understanding the development of RL models for the control of quadcopters are contextualized.

3.1. Quadcopter Dynamics

The dynamics of a quadcopter are characterized by its translational and rotational motion. The system state is represented by the vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \\ \boldsymbol{\theta} \\ \boldsymbol{\omega} \end{bmatrix},$$

where $\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ is the position, $\mathbf{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$ is the linear velocity, $\boldsymbol{\theta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ represents the orientation (to ll, pitch, yaw), and $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$ is the angular velocity.

The quadcopter's dynamics are governed by two key equations:

Translational Dynamics:

$$\dot{\mathbf{v}} = \frac{1}{m} \left(R(\boldsymbol{\theta}) \mathbf{F}_{\text{thrust}} + m\mathbf{g} \right),$$

where m is the mass, $\mathbf{g} = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$ m/s^2 is gravity, $R(\boldsymbol{\theta})$ is the rotation matrix that transforms thrust from the body frame to the world frame, and $\mathbf{F}_{\text{thrust}} = \begin{bmatrix} 0 & 0 & \sum_{i=1}^{4} T_i \end{bmatrix}^T$ is the thrust force.

Rotational Dynamics:

$$J\dot{\boldsymbol{\omega}} = \boldsymbol{\tau} - \boldsymbol{\omega} \times (J\boldsymbol{\omega}),$$

where J is the moment of inertia matrix and $\boldsymbol{\tau}$ is the torque vector, given by:

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} LC_b(T_2 - T_4) \\ LC_b(T_1 - T_3) \\ C_d(T_1 - T_2 + T_3 - T_4) \end{bmatrix}.$$

Here, L is the arm length, C_b is the thrust coefficient, and C_d is the drag coefficient.

These equations form the foundation for simulating quadcopter behavior [3], enabling control through the thrusts T_1, T_2, T_3, T_4 of the four motors.

3.2. PID Controllers

A Proportional-Integral-Derivative controller is a classical approach to controlling, where control inputs are generated based on the error $e(t)$ between the desired state $\mathbf{x}_{\text{ref}}(t)$ and the actual state $\mathbf{x}(t)$. The control law in the continuous-time domain is expressed as:

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{de(t)}{dt},$$

where K_P, K_I, and K_D are the proportional, integral, and derivative gains, respectively.

- **Proportional Term** ($K_P e(t)$): Reacts to the current error, providing immediate corrective action.
- **Integral Term** ($K_I \int_0^t e(\tau) d\tau$): Accumulates past errors, ensuring elimination of steady-state error.
- **Derivative Term** ($K_D \frac{de(t)}{dt}$): Predicts future error trends, reduce transient oscillations, and improving response stability.

In the simulation environments and experiments, discrete-time domain PID controllers are implemented, estimating the integral and derivative terms at each time step. Δt depends on the device's sampling frequency or the simulation setup. In this simulation, $Deltat$ is set to 0.01 second.

$$u[k] = K_P e[k] + K_I \sum_{i=0}^{k} e[i]\Delta t + K_D \frac{e[k] - e[k-1]}{\Delta t}$$

In quadcopter control, PID controllers are usually implemented to stabilize orientation (ϕ, θ, ψ) and regulate altitude (z) [10]. While easy to implement, PID controllers have difficulty dealing with nonlinear dynamics and unpredictable disturbances, motivating the exploration of RL-based methods [1].

3.3. Soft Actor-Critic (SAC)

Soft Actor-Critic is a model-free RL algorithm designed for continuous action spaces, combining features of both policy-based and value-based methods. SAC aims to optimize a stochastic policy $\pi_\theta(a \mid s)$ while encouraging exploration through entropy maximization. Its objectives are as follows [7]:

Policy Objective (Actor):

$$J_{\text{policy}}(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, a_t \sim \pi_\theta} \left[\alpha \, \mathcal{H}(\pi_\theta(\cdot|s_t)) - Q_\phi(s_t, a_t) \right],$$

where $\mathcal{H}(\pi_\theta)$ is the entropy of the policy, encouraging exploration, and $Q_\phi(s_t, a_t)$ is the action-value function, guiding the policy toward high-reward actions.

Q-Value Objective (Critic):

$$J_{\text{Q}}(\phi) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\phi(s_t, a_t) - y_{\text{target}} \right)^2 \right],$$

where the target y_{target} is computed as:

$$y_{\text{target}} = r_t + \gamma \, \mathbb{E}_{a_{t+1} \sim \pi} \left[Q_\phi(s_{t+1}, a_{t+1}) - \alpha \, \log \pi_\theta(a_{t+1}|s_{t+1}) \right].$$

Entropy Temperature Objective (α):

$$J_\alpha = \mathbb{E}_{a_t \sim \pi} \left[-\alpha \left(\log \pi_\theta(a_t|s_t) + \mathcal{H}_{\text{target}} \right) \right].$$

where $\mathcal{H}_{\text{target}}$ is a hyperparameter used to determine the desired amount of randomness. SAC is based on an actor-critic architecture: an actor learns the policy, while a critic estimates the Q-value. Being able to handle continuous action spaces and adapt to stochastic environments makes this algorithm a perfect candidate for quadcopter control, where precise adjustments in motor thrust are crucial, and the system must be robust to disturbances

SAC is based on the actor-critic architecture: an actor learns the policy, while two critics estimate the Q-value. All these features make this algorithm a perfect candidate for quadcopter control, especially with the focus on robustness to disturbances.[6].

Role of Two Q-Networks in SAC

The Soft Actor-Critic (SAC) algorithm utilizes two Q-networks, Q_{ϕ_1} and Q_{ϕ_2}, to estimate the expected return of a given state-action pair. These networks serve two key purposes:

1. **Variance Reduction through Double Q-Learning:** By using two Q-networks, SAC mitigates the overestimation bias. During the update step, the minimum of the two Q-values is used as the target, $Q_{\text{target}} = r + \gamma \left(\min \left(Q_{\phi_1}^{\text{target}}(s', a'), Q_{\phi_2}^{\text{target}}(s', a') \right) - \alpha \log \pi(a'|s') \right)$, where r is the reward, γ is the discount factor, α controls the entropy term, and s' and a' are the next state and action, respectively. This conservative estimation prevents the policy from exploiting overly optimistic Q-value predictions.
2. **Improved Stability and Robustness:** The redundancy provided by the two Q-networks enhances the robustness of the learning process. Even if one Q-network diverges temporarily due to noisy gradients or suboptimal updates, the other can compensate.

In practice, both Q_{ϕ_1} and Q_{ϕ_2} are updated to minimize the temporal difference error, while their target networks, $Q_{\phi_1}^{\text{target}}$ and $Q_{\phi_2}^{\text{target}}$, ensure the stability of target values during training by employing soft updates:

$$\phi^{\text{target}} \leftarrow \tau \phi + (1 - \tau) \phi^{\text{target}},$$

where τ is the soft update coefficient.

4. Literature Review

Using Soft Actor-Critic for Low-Level UAV Control [2] proposed a reinforcement learning-based framework that used the Soft Actor-Critic algorithm for low-level control of quadrotor UAVs. This work tries to overcome some of the major challenges of model-based control approaches, which rely on accurate mathematical models, by using a model-free reinforcement learning approach. The proposed framework integrates Coppelia Simulator with PyRep, thus enabling efficient training and reality transfer in high-dimensional, continuous action spaces. The maximum entropy framework of SAC was important for balancing exploration and exploitation. They also show a carefully designed reward function that worked to train a policy that is high-performing across a wide variety of scenarios, prioritizing positional accuracy and stability while penalizing angular velocities to minimize vibration. Empirical evaluations demonstrated that SAC was able to track stationary and dynamic targets for linear, square, and sinusoidal paths with much shorter training compared to previous methods. However, the authors noted limitations in predictive control for dynamic targets and suggested future research directions, including multi-objective reward functions and curriculum learning strategies to further enhance UAV control in complex environments.

This paper had a major influence on this project, inspiring the initialization parameters for the training environment and some of the evaluations performed. While we achieve a slower convergence to the target location when compared to the model proposed in this paper, we have eliminated the steady-state error those researchers found. We achieve a steady-state error of less than 1cm compared to the 20+cm from the results section of the attached paper.

Equivariant Reinforcement Learning for Quadrotor UAV [13] proposed an equivariant reinforcement learning framework for the efficiency of low-level control in quadrotors. The equivariance comes from a rotational symmetry inherent in quadrotor dynamics. The authors embed such symmetry into an actor-critic architecture, essentially reducing the effective dimensionality of the state space. This novelty provides much more efficient sampling and thus computational effectiveness in reinforcement learning algorithms such as TD3 and SAC. This guarantees that the value function remains unchanged and that the policy becomes quivalent under the state transformations. It also implies faster convergence and higher sample efficiency compared to its non-equivariant counterparts. Numerical simulations verified that the framework achieved both stabilization and tracking objectives, supported by a reward function balancing accuracy, stability, and smooth control.

5. Methodology

Overview

A simplistic simulation environment was initially constructed in Python to implement the quadcopter vehicle dynamics described above. The environment keeps track of the state, takes inputs from the four motor thrusts in the range of [0,1] that can be generated by the PID controller as well as the SAC algorithm, and outputs the new state of the simulated flight. To evaluate and compare the performance, both approaches will undergo five tests: point-to-point with and without wind noise, horizontal straight-line tracking, circular trajectory tracking, and a robustness drop test with an initial pitch angle of 45 degrees. One SAC-learned policy would be tested to complete the increasingly difficult flight tasks, ultimately proving its adaptability and robustness of quadcopter control.

PID controller setup

A classical cascaded PID controller with total of 18 tunable gain parameters is implemented. Figure 1 shows the structure of the controller. To verify the PID controller, the team successfully let the quadcopter lift to a specified z location while keeping the same ground coordinate [x,y]. After iterative adjustment of the gain parameters, the quadcopter can perform point-to-point movement that involves translational motion on all x, y, and z axes. The five tests completed by the PID controller will serve as a benchmark to compare their strengths and weaknesses in terms of positional accuracy and rate of recovery from disturbances.

SAC training

We settled on a few choices during environment development and training. For SAC we used a simple model comprised of 3 128-dimension linear layers with ReLU normalization, capped with separate final layers for mean action and standard deviation calculation, essentially introducing the exploration in the model architecture. The learning rates for the 3 networks are $\eta_{actor} = 3 \times 10^{-4}, \eta_{critics} = 1 \times 10^{-4}, \eta_{\alpha} = 1 \times 10^{-4}$. The initialization was done with different starting locations

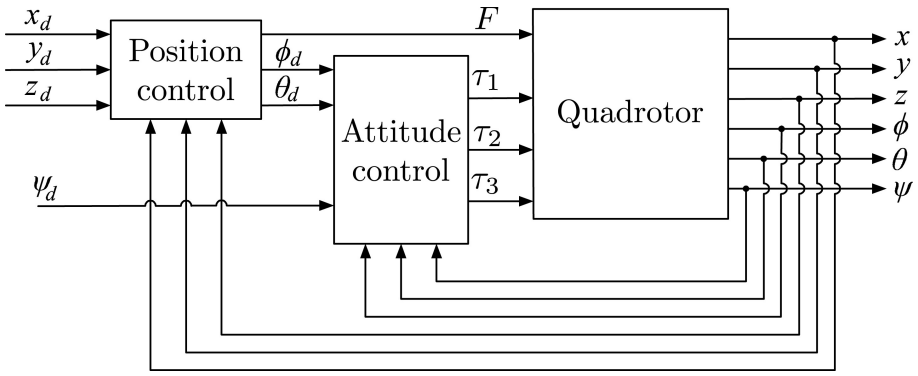

Figure 1: Cascaded PID quadcopter controller [9]

and orientations, and the reward function always incentivized reaching a stationary target located at $[0.0, 0.0, 1.7]$. The initialization can be described by:

- $[x, y]$ from Uniform distribution $\mathcal{U}(-1.5, -1.0, -0.5, 0.0, 0.5, 1.0, 1.5)$
- $[z]$ from $\mathcal{U}(1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2)$
- $[\phi, \theta, \psi]$ from $\mathcal{U}(-30.0, -25.0, -20.0, -15.0, -10.0, -5.0, 0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0)$
- Initial velocity and angular velocity were always initialized as $[0.0, 0.0, 0.0]$

Many different states and reward functions were tested. Ultimately, the state space $\mathcal{S} = \{$Position, Linear Velocity, Orientation, Angular Velocity$\}$, the exact 12 states introduced in quadcopter dynamics, achieved the best results. The action space is defined as the four motor thrusts $\mathcal{A} = \{T_1, T_2, T_3, T_4\}$, each action constrained in the range $[-1, 1]$. Discount factor $\gamma = 0.99$. The transition function for the quadcopter MDP is deterministic based on the vehicle dynamics. Although many reward functions achieved decent results, the best one used an exponentially decaying reward. This function outputs 10 when the quadcopter is 0 centimeter away from the target and 0 at infinity, combined with a sparse reward added to the when the quadcopter was within 1 cm of the target location.

It is worth mentioning that within the rewards tested, some included penalizing unwanted behavior like extreme angular velocities or motion away from the target. We also tried rewarding the correct direction of motion but nothing achieved the same results as the described function above. We also attempted to implement curriculum learning by using the already trained model and having it trained in an environment with moving targets, but this slightly worsened the model's performance on the tests described below.

6. Results

To evaluate the trained model we created 5 separate tests. The first test was comprised of a set of 100 random initializations similar to those performed during training, and the model's capability to reach a stationary target was evaluated. This task was the only one in scope during training and, as expected, the model completed all 100 trajectories, reaching within 5 cm of the target location and never causing early termination of the episodes. This is comparable to the results found by Barros and Colombini [2] that inspired this and another test.

However, as seen in Figure 2 our model achieves a significantly smaller steady-state error below 1 cm in most scenarios. Matching their initialization conditions of the plots shown we can see that both models reach their target location, and while theirs does so faster, it does so with a steady state error in the vertical axis of over 20 cm. It is worth noting that the simulation environment of the experiments is different both in terms of the dynamic calculations and the quadcopter properties which could cause the differences in velocity when reaching a target, but these don't explain the improvement in steady-state error leaving the improvements to the model. The PID performance for this test shown in Figure 7 also achieved accurate tracking in terms of z position with steady state error less than 2 cm, but struggled to converge to x, y target positions. Compared to SAC, the overshoot and oscillation in x, y position, roll, and pitch angle are significant partially

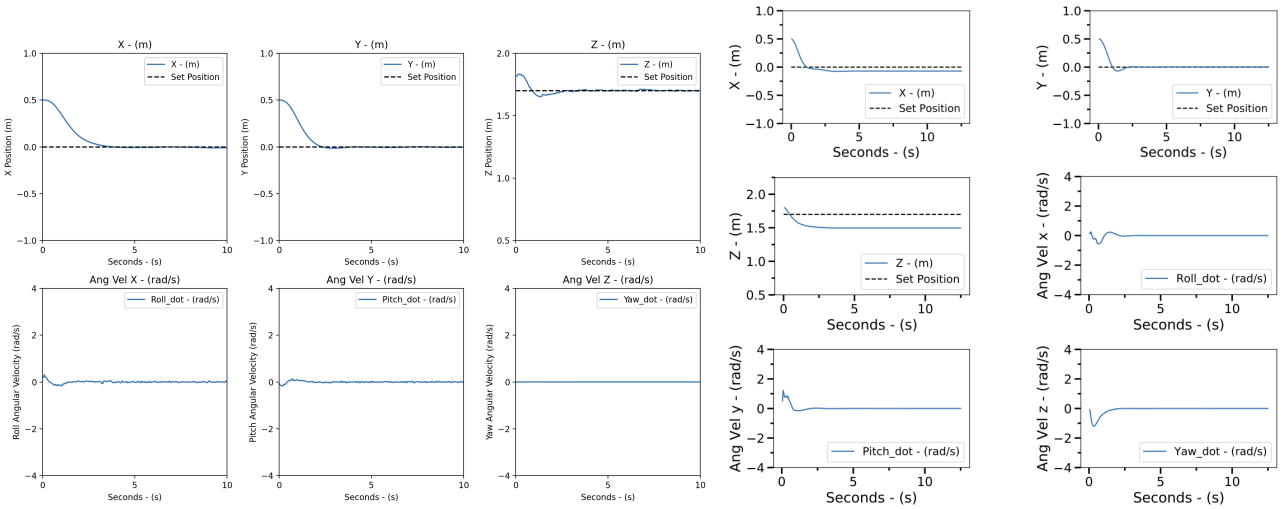

Figure 3: Drone Position (m) and Angular Velocities (rad/s). (a) x (b) y (c) z (d) $\dot{\phi}$ (e) $\dot{\theta}$ (f) $\dot{\psi}$.

Figure 2: Results comparison on reaching a stationary target for a) Our Implementation b) Barros and Colombini Implementation [2]

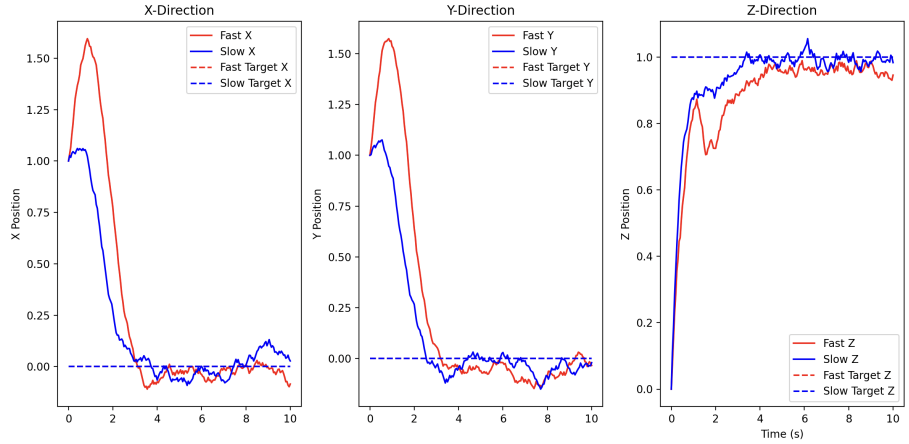

Figure 3: Results on reaching a stationary target with wind noise for fast wind (1m/s) and slow wind (0.2m/s)

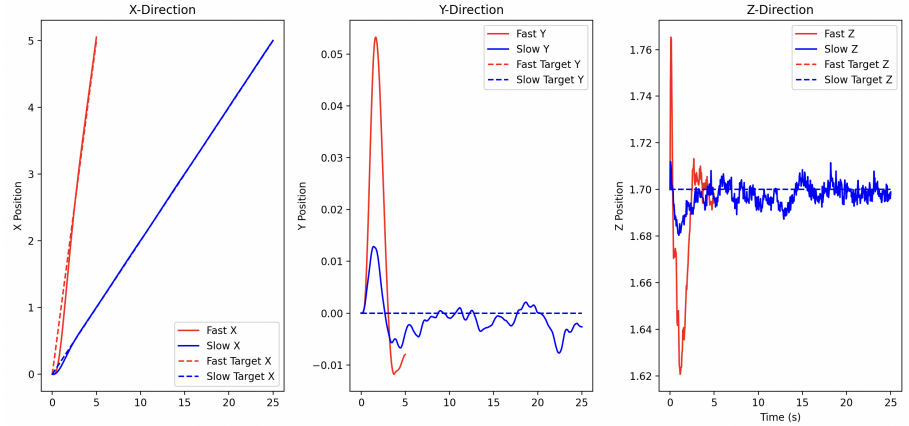

Figure 4: SAC on straight line path following at two speeds: Fast (1m/s) and Slow (0.2m/s)

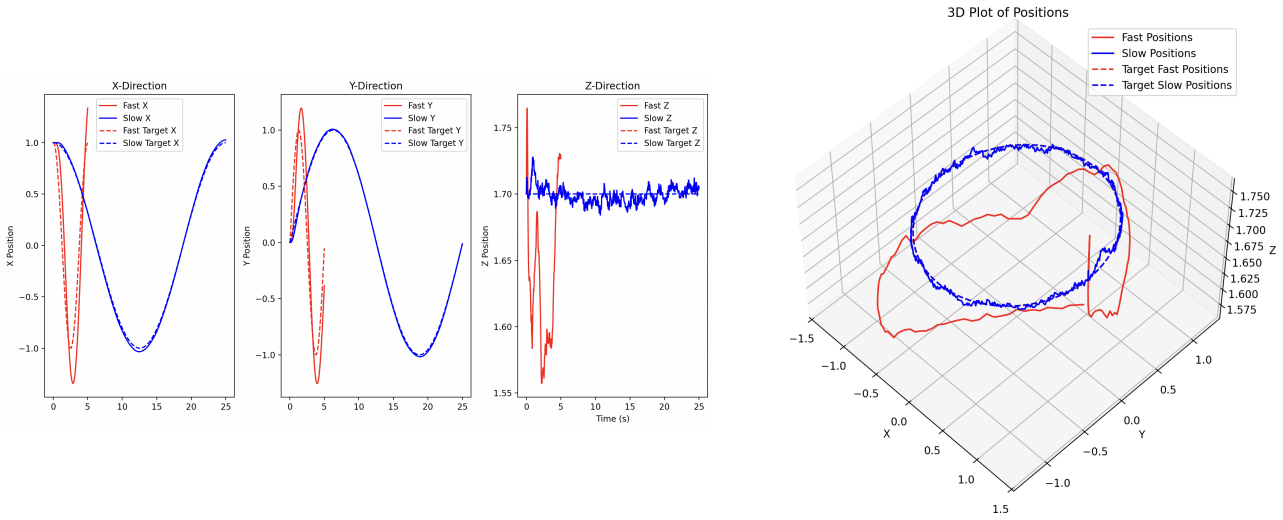

Figure 5: SAC circular path following at two speeds: Fast (1m/s) and Slow (0.2m/s)

because the PID parameters were not tuned as extensively as desired. The second test evaluates the same task of reaching a stationary target, but with wind noise introduced. The wind was calculated with by uniformly sampling from a distribution of $W_{\text{force}} = U(W_{\text{baseline}} \pm 0.4, 3)$ where U is the uniform distribution ranging from $W_{\text{baseline}} - 0.4$ to $W_{\text{baseline}} + 0.4$, applied as additional accelerations to the x, y, and z components. W_{baseline} is set to 1 m/s for the fast wind and 0.2 m/s for the slow wind. In Figure 3 we can see that even at the higher wind speed, the SAC model manages to reach the target, with a settling time around 4 seconds for all 3 axes. A significant undershoot occurs in the first 2 seconds as SAC attempted to react to the fast wind. While the steady-state error increases due to the noisy nature of the simulated wind, it is still relatively small when compared with the PID controller. As shown in Figure 8, at lower wind speed, the PID controller manages to track x, y target position while overshoots 8% on the z position and has a long settling time. At fast wind speed, the PID failed to maintain x,y position tracking but reaches the z target position with acceptable 14% overshoot.

The remaining 3 tests all evaluate the model's ability to generalize to unseen tasks [8]. The first 2 evaluate a path following ability on a straight and a circular path. The last one evaluates the models' ability to recover from extreme positions.

The straight-line and circular trajectory tests assess both approaches' ability to follow a predefined path under a fast speed that completes the path in 5 seconds, and a slow speed in 25 seconds. For the straight-line path that travels from x=0 to x=5 m, seen in Figure 4 and 9 the SAC model is extremely accurate at the slow speed achieving average steady-state errors below 1 cm. PID showed large deviation from the path in the first 5 seconds but achieved acceptable tracking capability. Although the errors for both approaches expectedly increased at faster speeds, PID was outperformed by SAC as PID struggled to match the straight-line at all time steps, but maintained more steady Y, Z position with smaller oscillations than SAC did.

Figures 5 and 10 demonstrate the capability of both controllers on tracking the circular path of 1 m diameter at 1.7 m height. At slow speed, the SAC accomplished exceptional accuracy in x, y location tracking and exhibited acceptable osccillation in z axis. The PID controller had overshoot at the convex and concave sections of x, y position curves, but kept perfect z position tracking. At faster speeds, the SAC started to deviate from the path and showed noticeable tracking error in z position. The PID performed worse in x, y position tracking, having difficulty in maneuvering at spots with rapid direction changes, but it managed to stay on the same z location throughout with minimal vibration.

Lastly, the SAC models' capabilities are significantly worse in the robustness section when the vehicle is initialized at 2 m high in z position and $45°$ in pitch angle with target position of [0,0,0], yet the SAC successfully settled to the goal position at the end of 5 seconds after oscillating substantially along x and z axes. This behavior was somewhat expected since all training scenarios were initialized between $-30°$ and $-30°$, meaning the model wasn't trained on the more extreme $45°$ angle and would likely fail at even larger angles. The PID controller surprisingly also completed the task with higher magnitude of oscillation in the x position and more error in z position than SAC. The PID exhibits lower overshoot in the z

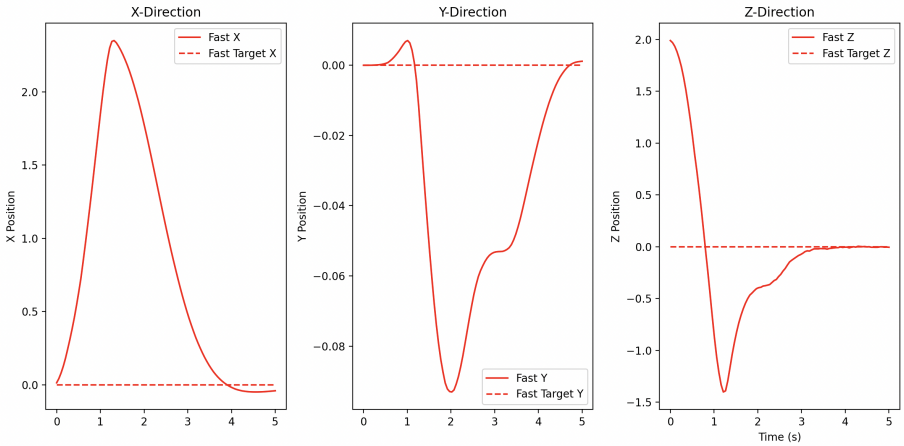

Figure 6: SAC reaching a stationary target when dropped 2 m above the target at a 45° angle

position than SAC.

SAC's struggle with unseen initial condition during training is also visible when testing the model extremely far from the target (10+ m). At these ranges the model doesn't seem to understand the task and simply moves in a random direction for the duration of the episode. We discuss further in the conclusion ideas that could improve the model in these edge cases and guide future research on the topic.

7. Conclusion

This project improved upon the work by Barros & Colombini [2] showing the feasibility of using the Soft Actor-Critic (SAC) reinforcement learning algorithm for quadcopter control. We've shown point-to-point navigation and stable hovering under various conditions, and demonstrated the model's generalization capabilities under a few tasks. Notably, our SAC model achieved a significantly smaller steady-state error when compared to existing benchmarks on reaching a stationary target and path following.

While the model worked well for most scenarios, it struggled with extreme initial conditions and far-reaching targets. Future work could involve increasing the training range, using curriculum learning to make the model more robust, and adding more disturbances to further challenge the model's adaptability.

A. PID Results And Plots

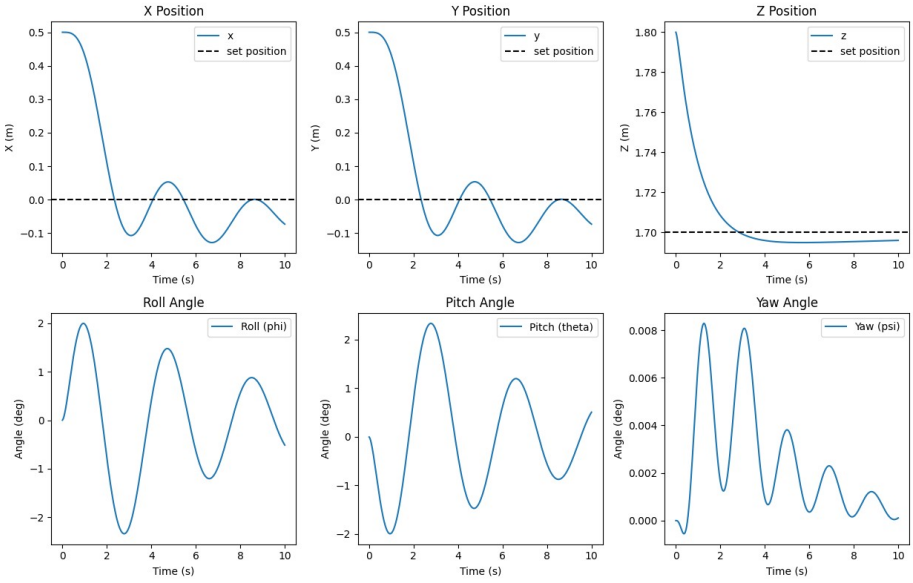

Figure 7: PID reaching a stationary target

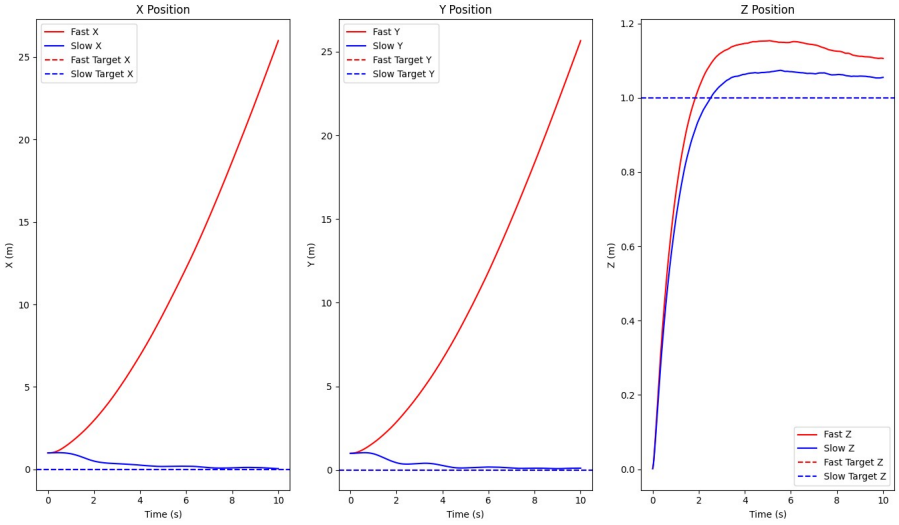

Figure 8: PID reaching a stationary target with wind noise for fast wind (1m/s) and slow wind (0.2m/s)

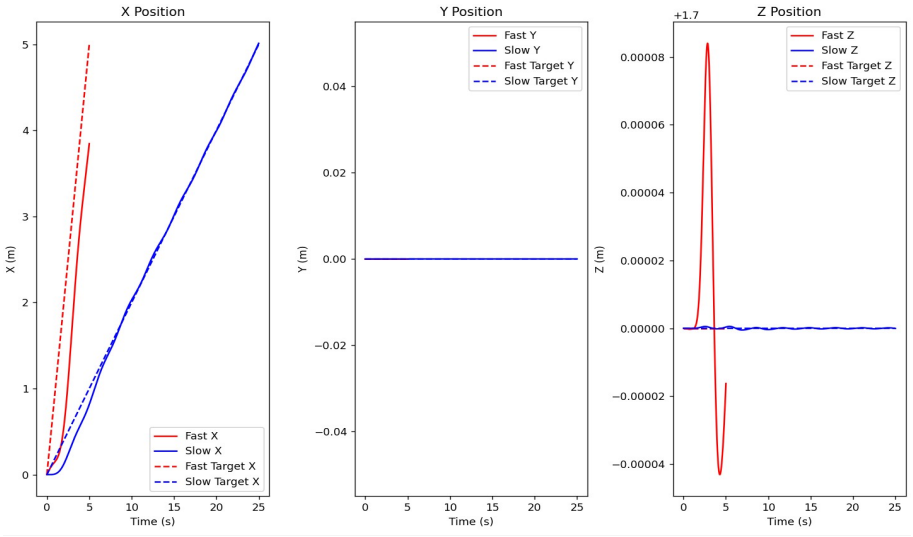

Figure 9: PID on straight line path following at two speeds: Fast (1m/s) and Slow (0.2m/s)

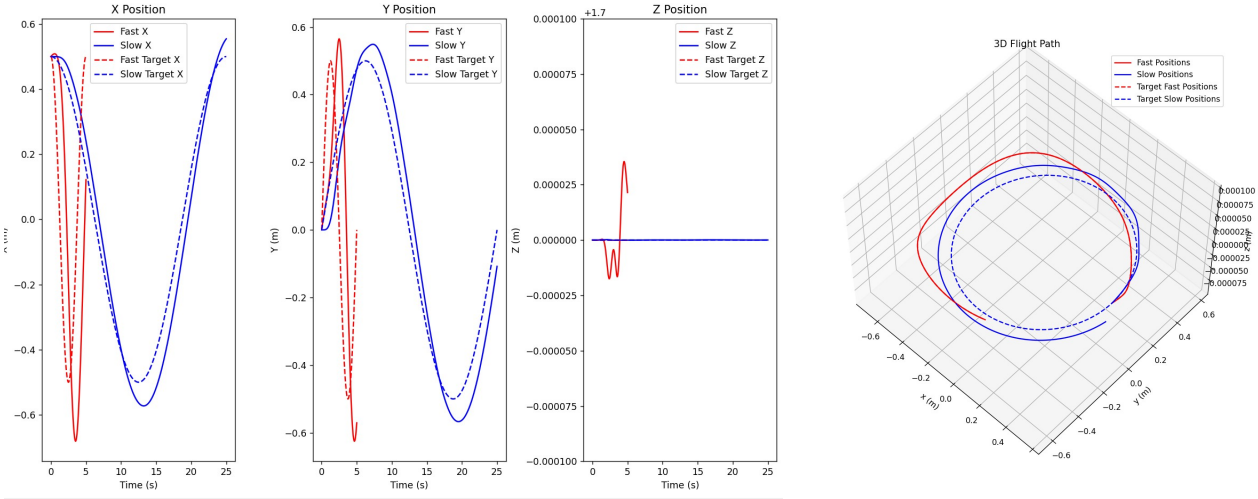

Figure 10: PID controller circular path following at two speeds: Fast (1m/s) and Slow (0.2m/s)

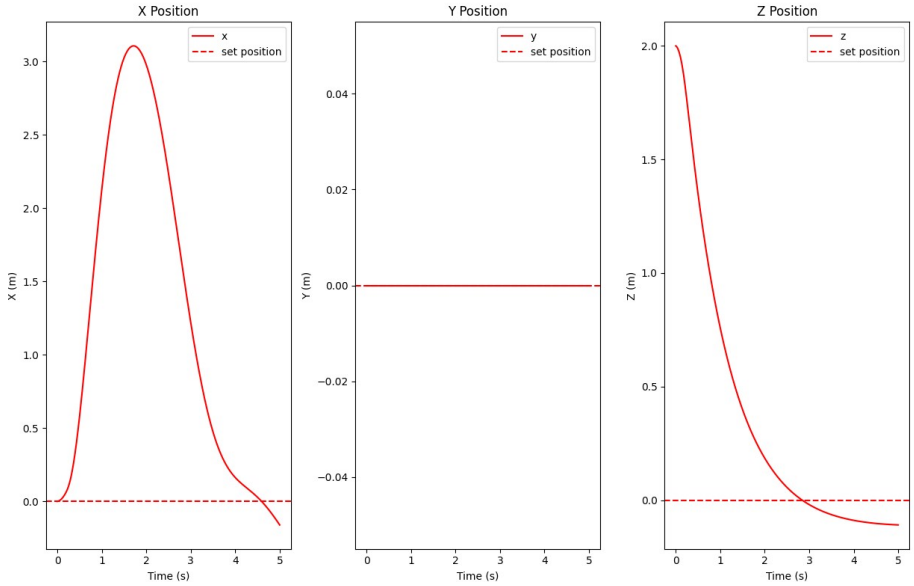

Figure 11: PID reaching a stationary target when dropped 2 m above the target at a 45° angle

References

[1] A. Baharuddin and M. A. M. Basri. Self-tuning pid controller for quadcopter using fuzzy logic. *International Journal of Robotics and Control Systems*, 2(1):1–10, 2022.

[2] G. M. Barros and E. L. Colombini. Using soft actor-critic for low-level uav control. *arXiv preprint arXiv:2010.02293*, 2020.

[3] T.-D. Do, N. X. Mung, and S. K. Hong. Deep reinforcement learning-based quadcopter controller: A practical approach and experiments. *arXiv preprint arXiv:2406.08815*, 2024.

[4] G. S. Eduardo and W. Caarls. Designing a robust low-level agnostic controller for a quadrotor with actor-critic reinforcement learning. *arXiv preprint arXiv:2210.02964*, 2022.

[5] S. Fang and Q. Zhu. Fundamental limits of controlled stochastic dynamical systems: An information-theoretic approach. *IEEE Transactions on Automatic Control*, 66(6):2563–2578, 2021.

[6] T. Haarnoja et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[7] T. Haarnoja et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1861–1870, 2018.

[8] G. Lopes, M. Ferreira, A. Simões, and E. Colombini. Intelligent control of a quadrotor with proximal policy optimization reinforcement learning. In *Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning*, pages 503–508, 11 2018.

[9] I. Lopez-Sanchez and J. Moreno-Valenzuela. Pid control of quadrotor uavs: A survey. *Annual Reviews in Control*, 56:100900, 2023.

[10] C. V. Nguyen and M. T. Nguyen. Design and simulation of pid-based control system for uav quadcopters. In *Advances in Information and Communication Technology*, pages 146–153. Springer, 2024.

[11] S. S. Sagar and Y.-G. Kim. Reinforcement learning-based drone simulators: survey, practice, and prospects. *Artificial Intelligence Review*, 56(5):3421–3450, 2023.

[12] I. Sharifi and A. Alasty. Self-tuning pid control via a hybrid actor-critic-based neural structure for quadcopter control. *arXiv preprint arXiv:2307.01312*, 2023.

[13] B. Yu and T. Lee. Equivariant reinforcement learning for quadrotor uav. *arXiv preprint arXiv:2206.01233*, 2022.