Quadcopter Hovering Control Using Reinforcement Learning in Isaac Lab

Wu Zining

Abstract

This project presents a reinforcement learning approach for robust quadcopter hovering control in simulated environments with wind disturbances. We explore and compare three prominent reinforcement learning algorithms: Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Deep Q-Network (DQN). We implement these algorithms in NVIDIA's Isaac Sim physics environment, enabling high-fidelity simulation of quadcopter dynamics. Our results demonstrate that the PPO algorithm achieves superior performance in maintaining stable hovering position despite wind disturbances, outperforming both TD3 and DQN approaches. The system successfully learns to control a Crazyflie quadcopter model, adjusting motor RPMs to counteract external disturbances and maintain a target hovering position. This research highlights the potential of reinforcement learning for developing robust control systems for unmanned aerial vehicles operating in challenging environments.

1 Introduction

Quadcopters have gained significant attention in recent years due to their versatility and applications in various domains including aerial photography, surveillance, delivery services, and search and rescue operations. A fundamental challenge in quadcopter operation is achieving stable hovering, especially in the presence of external disturbances such as wind. This capability is critical for precision tasks and serves as the foundation for more complex maneuvers.

Traditional control approaches for quadcopters typically rely on PID (Proportional-Integral-Derivative) controllers, LQR (Linear Quadratic Regulator), or MPC (Model Predictive Control). While these methods have proven effective in controlled environments, they often struggle with disturbances and uncertainties due to their reliance on accurate system models. Furthermore, tuning these controllers requires considerable expertise and time, making them less adaptable to changing conditions.

Reinforcement Learning (RL) offers a promising alternative that can potentially overcome these limitations. By learning from interaction with the environment, RL algorithms can develop control policies that are robust to disturbances without requiring explicit system models. This approach is particularly valuable for quadcopters operating in unpredictable outdoor environments where wind conditions can significantly impact flight stability.

In this project, we investigate the application of three prominent RL algorithms—PPO, TD3, and DQN—to the problem of quadcopter hovering control in the presence of wind disturbances. We utilize NVIDIA's Isaac Sim, a high-fidelity physics simulator, to create a realistic environment for training and evaluation. Our implementation is based on the Crazyflie quadcopter model, providing a practical platform for potential real-world transfer.

The key contributions of this work include:

- Implementation and comparison of PPO, TD3, and DQN algorithms for quadcopter hovering control
- Development of a realistic simulation environment with wind disturbance modeling
- Analysis of the performance and robustness of each algorithm under various conditions
- Insights into the design considerations for RL-based control systems for aerial robots

The remainder of this project is organized as follows: Section 2 provides an overview of related work in both traditional and learning-based approaches to quadcopter control. Section 3 details our methodology, including the reinforcement learning formulation, neural network architecture, and training process. Section 4 presents experimental results and comparisons between the algorithms. Section 5 discusses the advantages, limitations, and challenges of our approach. Finally, Section 6 concludes the project and outlines directions for future work.

2 Methodology

2.1 Problem Formulation

We formulate the quadcopter hovering problem as a reinforcement learning task where the agent must learn to maintain a stable position at a target height of 1 meter above the ground, while countering random wind disturbances. The RL framework comprises:

- **State space**: The state observed by the agent includes position (x, y, z), the target position, velocity components, attitude angles (roll, pitch, yaw), angular velocities, and position errors. The total state dimension is 18, providing a comprehensive representation of the quadcopter's dynamic state.
- **Action space**: The action space consists of the four motor RPM values, constrained between a minimum of 4500 RPM and a maximum of 5500 RPM.
- **Reward function**: The reward function is designed to encourage stable hovering by rewarding proximity to the target position and penalizing excessive movement and unstable attitudes. Specifically, it rewards reduced distance to the target position while penalizing high velocities and large tilt angles.
- **Environment dynamics**: The environment simulates realistic quadcopter physics including gravitational forces, propeller aerodynamics, and random wind disturbances. The wind model simulates varying force vectors with randomized magnitude and direction.

2.2 Simulation Environment

We developed our simulation environment using NVIDIA's Isaac Sim through the IsaacLab framework. IsaacLab is a higher-level abstraction layer built on top of Isaac Sim that provides simplified APIs and tools specifically designed for robotics research and reinforcement learning. While Isaac Sim provides the underlying high-fidelity physics simulation based on NVIDIA PhysX, IsaacLab offers convenient wrappers, pre-built robot models, and reinforcement learning utilities that accelerate development. Our simulation includes:

- A Crazyflie quadcopter model with realistic mass and inertia properties
- Precise propeller dynamics converting RPM to thrust
- Wind disturbance model generating random forces in the horizontal plane
- Realistic visualization capabilities for debugging and demonstration

The wind disturbance model generates random wind forces with magnitudes between 0.2 to 0.4 Newtons and randomly varying directions. The wind conditions are periodically updated to simulate changing wind patterns, creating a challenging control environment that requires adaptive behavior from the agent. IsaacLab's integration with Isaac Sim allowed us to efficiently implement these physics-based features while maintaining training speed suitable for reinforcement learning experiments.

2.3 Reinforcement Learning Algorithms

2.3.1 Proximal Policy Optimization (PPO)

PPO is our primary algorithm due to its stability and sample efficiency. PPO uses a trust region approach to policy optimization, limiting the size of policy updates to prevent performance collapse. The key components of our PPO implementation include:

- An actor-critic architecture with separate networks for policy and value estimation
- Clipped surrogate objective function to constrain policy updates
- Generalized Advantage Estimation (GAE) for variance reduction in policy gradient estimation
- Entropy regularization to encourage exploration

The actor network maps states to action distributions, parameterized by mean and standard deviation of a Gaussian distribution. The critic network estimates the value function to determine the advantage of each state-action pair. The networks share a similar structure but have separate parameters:

The actor and critic networks each use a two-layer architecture with 256 neurons per hidden layer and ReLU activations. The actor outputs the mean and log standard deviation of the action distribution, while the critic outputs a single value estimate.

Algorithm 1 PPO Training Algorithm

```
1: Initialize actor network $\pi_\theta$ and critic network $V_\phi$
2: for episode = 1 to max_episodes do
3:     Reset environment, get initial state $s_0$
4:     episode_reward $\leftarrow$ 0
5:     for t = 0 to max_steps do
6:         Sample action $a_t \sim \pi_\theta(a_t|s_t)$
7:         Execute $a_t$, observe $s_{t+1}$, reward $r_t$, done signal
8:         Store $(s_t, a_t, r_t, s_{t+1}, \log \pi_\theta(a_t|s_t), V_\phi(s_t))$ in buffer
9:         episode_reward $\leftarrow$ episode_reward + $r_t$
10:        if buffer is full or episode ends then
11:            Compute returns and advantages using GAE
12:            for k = 1 to ppo_epochs do
13:                Sample mini-batches from buffer
14:                Compute PPO loss:
15:                $L_{CLIP}(\theta) = \hat{\mathbb{E}}_t[\text{min}(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$
16:                $L_{VF}(\phi) = \hat{\mathbb{E}}_t[(V_\phi(s_t) - R_t)^2]$
17:                $L_S(\theta) = \hat{\mathbb{E}}_t[S[\pi_\theta](s_t)]$
18:                $L = -L_{CLIP}(\theta) + c_1 L_{VF}(\phi) - c_2 L_S(\theta)$
19:                Update $\theta$ and $\phi$ by gradient descent on $L$
20:            end for
21:            Clear buffer
22:        end if
23:        if done then
24:            Break
25:        end if
26:    end for
27: end for
```

2.3.2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is an off-policy algorithm designed to address function approximation errors in actor-critic methods. Our TD3 implementation includes:

- Twin critics to reduce overestimation bias
- Delayed policy updates to allow the critics to converge
- Target policy smoothing through noise addition to the target actions
- Experience replay buffer for sample efficiency

We implemented TD3 using the Stable Baselines3 library, which provides a robust implementation with proven performance across various control tasks.

2.3.3 Deep Q-Network (DQN)

While DQN is traditionally used for discrete action spaces, we adapted it for our continuous control problem by discretizing the action space. The key components include:

- Action space discretization into nine discrete actions per motor
- Experience replay buffer to break correlations between consecutive samples
- Target network for stable learning
- Epsilon-greedy exploration strategy

The DQN network consists of three fully connected layers with ReLU activations, mapping state inputs to Q-values for each possible action.

2.4 Training Process

The training process for each algorithm follows a similar structure:

1. Initialize the environment with random starting conditions
2. Execute the current policy and collect experience
3. Update the policy based on the collected experience
4. Evaluate the updated policy periodically
5. Save the best-performing model based on evaluation metrics

Table 1: Hyperparameters for the reinforcement learning algorithms

Hyperparameter	**PPO**	**TD3**	**DQN**
Learning rate	3e-4 (actor), 1e-3 (critic)	1e-4	5e-4
Discount factor	0.99	0.99	0.99
Buffer size	4,096	20,000	50,000
Batch size	128	256	64
Number of epochs	10	–	–
GAE parameter	0.95	–	–
Clip ratio	0.2	–	–
Entropy coefficient	0.01	–	–
Value function coefficient	0.5	–	–
Policy delay	–	3	–
Target policy noise	–	0.1	–
Noise clip	–	0.3	–
Target network update frequency	–	–	5,000 steps
Initial exploration	–	–	1.0
Final exploration	–	–	0.1
Exploration fraction	–	–	0.5
Training episodes	2,000	2,000	2,000

Each algorithm was trained for a different number of episodes as indicated in Table 1, providing sufficient learning time for fair comparison while accounting for their different sample efficiencies.

3 Results

3.1 Training Performance

Figure 1: Learning curves showing average episode rewards during training for PPO, TD3, and DQN algorithms. PPO demonstrates successful convergence while TD3 and DQN failed to learn effective hovering policies.

Figure 1 illustrates the training progress of the three algorithms in terms of average episode reward. Our experiments revealed significant differences in learning capabilities among the algorithms. PPO successfully learned to stabilize the quadcopter, achieving stable hovering within approximately 500 episodes. In contrast, both TD3 and DQN failed to learn effective hovering policies despite extended training periods.

TD3 showed the poorest performance, with the quadcopter unable to maintain even basic horizontal stability. The agent repeatedly failed to learn appropriate motor control patterns, resulting in unstable flight behavior and rapid crashes. DQN performed slightly better, managing to control the quadcopter for a controlled descent in no-wind conditions, but failed to maintain the target hover position. The discrete action space likely contributed to DQN's limited success, as it restricted the fine-grained control necessary for stable hovering.

The superior performance of PPO can be attributed to its on-policy learning approach, which provides more stable updates, and its clipped objective function, which prevents destructively large policy updates. These features proved critical for the quadcopter hovering task, where precise, coordinated control actions are essential.

3.2 Evaluation in Wind Conditions

To evaluate the robustness of the successful PPO policy, we tested it in various wind conditions:

- No wind: Baseline condition to assess general performance
- Light wind: Wind forces of 0.1-0.2 N
- Medium wind: Wind forces of 0.3-0.5 N
- Strong wind: Wind forces of 0.6-0.8 N
- Varying wind: Randomly changing wind direction and strength

For each condition, we measured:

- Position error: Average Euclidean distance from the target position
- Hover success rate: Percentage of time the quadcopter maintained position within 0.1m of the target
- Flight duration: Time until failure or maximum test duration

Table 2: Performance of PPO in different wind conditions

Wind Condition	Position Error (m)	Hover Success Rate (%)	Flight Duration (s)
No wind	0.03 ± 0.01	98.7	>30
Light wind	0.05 ± 0.02	95.8	>30
Medium wind	0.08 ± 0.03	92.5	>30
Strong wind	0.14 ± 0.06	84.1	28.6
Varying wind	0.11 ± 0.05	87.3	>30

Table 2 summarizes the performance metrics for the PPO algorithm under various wind conditions. The PPO policy demonstrated excellent robustness, maintaining close proximity to the target position even in challenging wind conditions. In no-wind and light wind conditions, the quadcopter maintained position with minimal error. Even in medium and varying wind conditions, the hover success rate remained above 85%. Only under strong wind conditions did we observe occasional instability, though the quadcopter could still maintain flight for extended periods.

3.3 Qualitative Analysis

Qualitative observation of the PPO flight behavior reveals important characteristics:

- **Adaptability**: The PPO policy demonstrates excellent adaptability to changing wind conditions, quickly adjusting motor RPMs to counteract disturbances as they occur.
- **Smooth control**: The policy produces smooth control actions with balanced, coordinated adjustments across all four motors. This prevents erratic movements and contributes to stable flight.
- **Recovery capability**: When displaced from the target position by strong wind gusts, the quadcopter efficiently returns to the hover position without overshooting or oscillation.
- **Attitude stability**: The policy effectively maintains level attitude despite lateral wind forces, keeping roll and pitch angles within ±5 degrees during most flight conditions.

Figure 2: Example trajectories of quadcopter position under varying wind conditions using the PPO controller. The plot shows position over time with colored sections indicating different wind strengths.

Figure 2 illustrates typical flight trajectories for the PPO controller under varying wind conditions. The quadcopter maintains a tight clustering around the target position (0,0,1) during no-wind and light wind conditions, with slightly increased dispersion as wind strength increases. Even during transition between different wind conditions, the controller rapidly adapts to maintain stability.

3.4 Motor RPM Analysis

Analysis of the motor RPM patterns provides insight into how the PPO policy adapts to wind disturbances:

- **Differential thrust**: When wind comes from a particular direction, the policy increases RPM for motors on the windward side while maintaining overall thrust balance. This creates a controlled tilt to counteract the lateral wind force.
- **Coordinated response**: All four motors show coordinated adjustments, with complementary patterns that maintain attitude stability while counteracting disturbances.
- **Anticipatory behavior**: Interestingly, the policy appears to develop anticipatory behavior after extended training, making small preventative adjustments that improve resistance to recurring wind patterns.

Figure 3: Motor RPM patterns of the PPO controller responding to a wind gust from the +X direction. The plot shows the differential response of each motor to maintain stability.

4 Discussion

4.1 Advantages and Limitations

4.1.1 Advantages

The PPO-based approach demonstrated several advantages for quadcopter hovering control:

- **Robustness to disturbances**: The learned policy effectively counters wind disturbances without explicit modeling of the disturbance forces.
- **Adaptive behavior**: Unlike fixed-gain controllers, the RL-based approach adapts its control strategy based on the observed state, leading to more effective disturbance rejection.
- **No system identification required**: The approach does not require detailed system identification or explicit modeling of quadcopter dynamics, making it more generalizable.
- **End-to-end learning**: The direct mapping from sensor inputs to motor commands eliminates the need for separate state estimation and control layers.

4.1.2 Limitations

Despite its strong performance, our approach has several limitations:

- **Sample inefficiency**: RL algorithms require significant training data, which can be expensive to collect in real-world systems.
- **Reality gap**: Policies trained in simulation may not transfer perfectly to real hardware due to discrepancies in dynamics and sensor characteristics.
- **Limited generalization**: The current policy is optimized for hovering at a specific height; generalizing to arbitrary setpoints would require additional training or modifications.
- **Computational requirements**: Training effective policies requires substantial computational resources, particularly for high-dimensional state spaces.

4.2 Algorithm Comparison

Our exploration of PPO, TD3, and DQN for quadcopter hovering control revealed significant differences in their applicability to this challenging task:

- **Training progression**: Our development process involved first attempting to use TD3, then DQN, before finally achieving success with PPO. This trajectory offers valuable insights into the relative strengths of each algorithm for this specific control problem.

- **Policy stability**: TD3 showed the poorest performance, with the quadcopter unable to maintain even basic horizontal stability. Despite extensive hyperparameter tuning, TD3 failed to learn an effective control policy for hovering.
- **Action space considerations**: DQN performed slightly better than TD3, managing controlled descent in no-wind conditions. However, the discretized action space proved too limiting for the precise, continuous control needed for stable hovering.
- **Sample efficiency vs. stability**: While off-policy methods like TD3 and DQN theoretically offer better sample efficiency, the on-policy PPO algorithm provided the stability necessary for learning in this complex, dynamic environment.

PPO's success can be attributed to its policy clipping mechanism, which prevents large, destabilizing policy updates, and its on-policy learning approach that allows it to adapt more effectively to the quadcopter's complex dynamics. These features proved essential for the quadcopter hovering task, where precise, coordinated control actions and stability under varying conditions are critical requirements.

4.3 Challenges and Lessons Learned

Throughout this project, we encountered several challenges and learned valuable lessons:

- **Algorithm selection**: The most significant lesson was the importance of algorithm selection for specific control tasks. Our sequential attempts with TD3, DQN, and finally PPO demonstrated that theoretical advantages of certain algorithms may not translate to practical performance in complex control scenarios.
- **Reward function design**: Crafting an effective reward function proved challenging. Initial attempts with sparse rewards led to poor learning, while overly complex rewards caused conflicting objectives. We found that a balanced approach combining position error, attitude stability, and smooth control actions worked best.
- **Training progression**: We discovered the value of incremental training, starting with simpler scenarios (no wind) before progressing to more complex ones (varying wind). This approach allowed the policy to learn fundamental control first before adapting to disturbances.
- **Hyperparameter sensitivity**: RL algorithms are sensitive to hyperparameter choices. Extensive tuning was required to achieve stable learning, particularly for the learning rates, batch sizes, and policy update frequencies. This sensitivity varied across algorithms, with PPO showing greater robustness to hyperparameter settings.
- **Simulation challenges**: Creating a realistic wind model required careful calibration to ensure the disturbances were challenging but not impossible to overcome. Additionally, ensuring consistent simulation speed across experiments was important for fair comparison.

4.4 Future Work

Based on our findings, several promising directions for future work emerge:

- **Sim-to-real transfer**: Investigating techniques to bridge the reality gap and deploy the successful PPO policy on real quadcopter hardware.
- **Multi-task learning**: Extending the approach to handle multiple tasks such as hovering, trajectory following, and obstacle avoidance within a single policy.
- **Hybrid approaches**: Combining PPO with traditional control techniques to leverage the strengths of both paradigms, potentially using a model-based approach for basic stability and RL for adaptive disturbance rejection.
- **Meta-learning**: Developing meta-learning approaches that can quickly adapt to new conditions, such as changing payloads or mechanical wear.
- **Alternative algorithms**: Exploring other on-policy methods such as Soft Actor-Critic (SAC) that might offer improved sample efficiency while maintaining the stability benefits demonstrated by PPO.

5 Conclusion

In this project, we presented a reinforcement learning approach for robust quadcopter hovering control in the presence of wind disturbances. Through systematic exploration, we found that PPO achieved successful performance in terms of position accuracy and hover stability, while both TD3 and DQN failed to learn effective control policies despite extensive tuning efforts.

Our development process, which involved sequential attempts with different algorithms, provides valuable insights into the relative strengths of these methods for complex control tasks. The successful PPO policy demonstrated remarkable robustness to varying wind conditions, maintaining stable hovering even under significant disturbances.

The results demonstrated that RL-based approaches, specifically on-policy methods like PPO, can effectively learn to stabilize a quadcopter without explicit system modeling or disturbance estimation. The trained policy showed adaptability to varying wind conditions, highlighting the potential of RL for developing robust control systems.

The key insights from our work include:

- The on-policy nature and policy clipping mechanism of PPO proved critical for successful learning in this complex control task
- The sequential training approach (first no-wind, then wind conditions) facilitated more effective policy development
- Careful reward function design balancing position accuracy, attitude stability, and control smoothness was essential

Despite the promising results, several challenges remain for real-world deployment, including the reality gap between simulation and hardware, computational requirements for training, and limited generalization to new tasks. Future work will focus on addressing these limitations through sim-to-real transfer techniques and more generalizable learning frameworks.

We believe this work contributes valuable insights to the field of quadcopter control, demonstrating both the potential and challenges of reinforcement learning approaches. With continued advancement, RL-based control systems hold significant promise for enabling robust autonomous operation of quadcopters in challenging real-world environments.

Acknowledgements

This research utilized code partially generated by artificial intelligence, including basic reinforcement learning algorithm implementations and environment interaction interfaces. These generated codes were subsequently deeply debugged and optimized by the research team to ensure their effectiveness and stability in quadcopter control tasks. AI-assisted programming significantly accelerated our development process, allowing us to iterate and test different algorithm designs more quickly.

We especially thank the development team of the Isaac Lab simulation environment, whose high-fidelity physics simulation platform provided critical support for this research. We also appreciate all laboratory members for their valuable suggestions and technical support during algorithm debugging and experimental design.

References

[1] Bouabdallah, S., Noth, A., & Siegwart, R. (2004). PID vs LQ control techniques applied to an indoor micro quadrotor. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

[2] Chitsaz, H., & LaValle, S. M. (2011). Time-optimal paths for a Dubins airplane. In IEEE Conference on Decision and Control.

[3] Alexis, K., Nikolakopoulos, G., & Tzes, A. (2011). Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. Control Engineering Practice.

[4] Madani, T., & Benallegue, A. (2006). Backstepping control for a quadrotor helicopter. In IEEE/RSJ International Conference on Intelligent Robots and Systems.

[5] Xu, R., & Özgüner, Ü. (2006). Sliding mode control of a quadrotor helicopter. In Proceedings of the IEEE Conference on Decision and Control.

[6] Balas, C., & Balas, M. J. (2018). Disturbance accommodating adaptive control with application to wind turbines. In American Control Conference.

[7] Wang, C., Song, B., Huang, P., & Tang, C. (2016). Trajectory tracking control for quadrotor robot subject to payload variation and wind gust disturbance. Journal of Intelligent & Robotic Systems.

[8] Zhang, T., Kahn, G., Levine, S., & Abbeel, P. (2016). Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In IEEE International Conference on Robotics and Automation.

[9] Hwangbo, J., Sa, I., Siegwart, R., & Hutter, M. (2017). Control of a quadrotor with reinforcement learning. IEEE Robotics and Automation Letters.

[10] Ha, J. S., Park, H. J., & Choi, H. L. (2020). Sample-efficient reinforcement learning for position control of a quadrotor. In IEEE/RSJ International Conference on Intelligent Robots and Systems.

[11] Koch, W., Mancuso, R., West, R., & Bestavros, A. (2019). Reinforcement learning for UAV attitude control. ACM Transactions on Cyber-Physical Systems.

[12] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. Science Robotics.

[13] Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In International Conference on Machine Learning.

[14] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. Neural Computation.