ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Department of Electrical, Electronic, and Information Engineering "Guglielmo Marconi" - DEI

Second Cycle Degree/Two Year Master in Automation Engineering

Towards Reinforcement Learning Control of an Aerial Robot and Physical Interaction Tasks

Supervisor:
Prof. Gianluca Palli

Defended By:
Mayuresh Pachore

International Supervisor:
Ayham Alharbat, MSc

Graduation Session March 2026
Academic Year 2024/2025

Abstract

Aerial robots are increasingly used for applications such as infrastructure inspection, environmental monitoring, and disaster response. While most systems operate in free flight, many emerging tasks require aerial robots to physically interact with their environment. Such interaction introduces significant challenges, as contact forces can destabilize the aerial platform and require precise control of motion and force.

This thesis investigates the use of reinforcement learning (RL) for controlling a fully actuated aerial robot with the long-term goal of enabling aerial physical interaction tasks. A reinforcement learning framework is developed in which the control policy is trained in simulation using a progressive learning strategy. The training process begins with velocity stabilization and gradually extends to altitude control, position tracking, and stable hovering.

The proposed approach is implemented using physics-based simulation environments that model the robot dynamics, actuator behavior, and contact interactions. The results demonstrate that reinforcement learning can successfully learn stable hovering control for a fully actuated aerial platform. These results establish the essential components required for future research on aerial physical interaction.

Keywords: Aerial Robotics, Reinforcement Learning, Fully Actuated Aerial Robot, Hovering Control, Aerial Physical Interaction, Robot Control, MuJoCo, BRAX, Simulation-based learning

Riassunto

I robot aerei sono sempre più utilizzati in applicazioni come l'ispezione di infrastrutture, il monitoraggio ambientale e la risposta ai disastri. Sebbene la maggior parte dei sistemi operi in volo libero, molte applicazioni emergenti richiedono che i robot aerei interagiscano fisicamente con l'ambiente. Tali interazioni introducono sfide significative, poiché le forze di contatto possono destabilizzare la piattaforma aerea e richiedono un controllo preciso del movimento e delle forze.

Questa tesi studia l'utilizzo del reinforcement learning (RL) per il controllo di un robot aereo completamente attuato con l'obiettivo a lungo termine di abilitare compiti di interazione fisica aerea. Viene sviluppato un framework di reinforcement learning in cui la politica di controllo viene addestrata in simulazione attraverso una strategia di apprendimento progressiva. Il processo di addestramento parte dalla stabilizzazione della velocità e viene gradualmente esteso al controllo dell'altitudine, al tracciamento della posizione e al mantenimento dell'hovering stabile.

L'approccio proposto è implementato utilizzando ambienti di simulazione basati su modelli fisici che rappresentano la dinamica del robot, il comportamento degli attuatori e le interazioni di contatto. I risultati dimostrano che il reinforcement learning è in grado di apprendere un controllo di hovering stabile per una piattaforma aerea completamente attuata. Questi risultati forniscono le basi necessarie per future ricerche sull'interazione fisica aerea.

Parole chiave: Robotica Aerea, Reinforcement Learning, Robot Aereo Completamente Attuato, Controllo di Hovering, Interazione Fisica Aerea, Controllo Robotico, MuJoCo, BRAX, Apprendimento Basato Sulla Simulazione

Acknowledgment

I would first like to express my sincere gratitude to my supervisor, Professor Gianluca Palli, for his guidance, support, and trust throughout the course of this thesis. His insight, encouragement, and dedication to research have been a constant source of motivation, and I am deeply grateful for the opportunity to work under his supervision.

I am also very thankful for the opportunity to conduct my thesis research abroad at the Smart Mechatronics and RoboTics (SMART) Research Group. Being part of such a stimulating and collaborative research environment has been an invaluable experience both academically and personally. I would like to thank the entire laboratory for welcoming me and for creating an atmosphere that encouraged learning, curiosity, and collaboration.

My special thanks go to my co-supervisor, Ayham Alharbat, MSc, whose guidance, availability, and thoughtful discussions were essential throughout this work. His advice and support greatly contributed to the development of this thesis.

I would also like to thank Professor Abeje Y. Mersha, Kousheek Chakraborty, MSc, Benjamin van Manen, MSc, and Gerdine Meijer, as well as the other members of the SMART Research Group, for their support, discussions, and collaboration which made my time in the laboratory both productive and enjoyable.

I am deeply grateful to Professor S. S. Shirguppikar, who during my bachelor's studies constantly pushed me beyond my financial, mental, and academic limitations. Without his encouragement and belief in me, I would have never imagined pursuing a master's degree abroad. His guidance played a major role in bringing me to the University of Bologna, and for that I will always remain thankful.

I would also like to thank Professor Alessandro Macchelli for giving me the opportunity to participate in ERF 2024, which marked an important milestone in my academic journey. I am also grateful to my colleagues Giovanna, Andrea, Emanuele, Francesco, Lucrezia, and Davide for their support and collaboration during this conference experience.

My master's journey in Italy would not have been the same without the amazing people I met along the way. Martina, thank you for sharing your Industrial Robotics notes and for all the moments we spent preparing for exams together.

Francesco, thank you for introducing me to the real experience of Italy and for the hospitality you and your family showed me. Spending time with you and Zak created memories I will always cherish.

Gabriele, thank you for constantly pushing me to aim higher and go beyond my limits. Your encouragement played a significant role in my participation in ERF 2024 and in my decision to pursue my thesis in the Netherlands.

I would also like to thank my former roommates Marco, Guglielmo, and Daniel. The time spent together in Ergo accommodation remains unforgettable—from learning Neapolitan card games to sharing food, jokes, and everyday experiences that made my early days in Italy special.

Guicy, an Italian more indian then me, many of my best memories in Italy are because of you. I am grateful to have found such a genuine friend during my master's journey.

As an international student, I would also like to thank my friends Ammar, Messam, Sammad, and Halil. Your help during courses, discussions during stressful moments, and the friendships we built made this journey much easier and more enjoyable.

To my friends Mayank, Shreyas, Mahesh, Aditya, and Shruti, thank you for creating a sense of home away from home. Being around you always made me feel comfortable and supported.

I would also like to thank Om and Pruthvi for their constant encouragement and motivation during some of the most difficult moments of my master's journey. Their support helped me stay focused and continue moving forward.

After losing my scholarship, I started working part-time at RGIS. I thank Swathi for encouraging me during that period and for the memorable late-night bicycle rides and shared experiences that made that time more meaningful.

I am also grateful to Kumar and Nik for their kindness and support during difficult moments, welcoming me as if I were a member of their home.

Vaishali, thank you for always being someone I could rely on whenever I needed a break from reality and for constantly reminding me to stay positive.

Christy, despite our jokes and arguments, your care and support meant a lot to me, and I will always remember your kindness.

Tara, thank you for being a great friend and for the many walks and conversations that helped me stay calm and positive during difficult times.

A special thanks goes to Santosh. From my bachelor's years to my master's journey, your friendship and support have been invaluable. Without you, this journey would not have been the same, and in many ways this achievement belongs to you as well. Basically, you have earned two master's degrees today

Finally, I would like to thank my family—my grandparents, my parents, and my brother. Their love, sacrifices, and constant belief in me made everything possible. This master's degree belongs to them as much as it belongs to me.

Thank you all.

Contents

List of Figures

List of Tables

Nomenclature

Symbol	Description
m	Mass of the aerial robot
J	Inertia matrix of the aerial robot
I_3	3×3 identity matrix
p	Position of the aerial robot in the world frame
$\dot{p}$	Linear velocity of the aerial robot
$\ddot{p}$	Linear acceleration of the aerial robot
p_E^B	Position of the end–effector in the body frame
p_E	End–effector position
R_B	Rotation matrix of the body frame with respect to the world frame
R_E	Rotation matrix of the end–effector frame
R_d	Desired rotation matrix
ω	Angular velocity of the aerial robot
$\dot{\omega}$	Angular acceleration of the aerial robot
u	Rotor thrust input vector
u_{ref}	Reference rotor thrust command
G_f	Force allocation matrix
G_τ	Moment allocation matrix
G	Control allocation matrix
f_c	Contact force applied at the end–effector
$f_{c,z}$	Contact force along the surface normal
f_e	External disturbance force
τ_c	Torque generated by contact interaction
τ_e	External disturbance torque
g	Gravitational acceleration
o_t	Observation vector of the RL agent
s_t	State of the environment
a_t	Action generated by the policy
π_θ	Reinforcement learning policy
p_E^{ref}	Desired end–effector position
f^{ref}	Desired contact force
r	Reward function
γ	Discount factor

Symbol	Description
r_{pos}	Position tracking reward
r_{pos_fine}	Fine position reward
r_{force}	Force tracking reward
r_{act}	Action smoothness reward
r_{flat}	Orientation alignment reward
r_{term}	Termination penalty
λ_i	Reward weighting coefficients
σ	Reward scaling parameter

Chapter 1

Introduction

Aerial robotics has experienced rapid growth over the past decade, driven by advances in sensing, onboard computation, and autonomous control algorithms. Unmanned Aerial Vehicles (UAVs), particularly multirotor platforms, have become widely used for applications such as infrastructure inspection, environmental monitoring, disaster response, and aerial mapping. Their ability to perform vertical take-off and landing, hover in place, and maneuver in confined environments makes them highly versatile robotic systems.

Traditionally, aerial robots have primarily been used for perception-oriented tasks where the vehicle observes the environment without establishing physical contact. However, many emerging real-world applications require aerial robots to interact directly with their surroundings. Examples include valve turning, structural inspection through contact sensing, cleaning or maintenance of industrial infrastructure, and manipulation tasks in hazardous environments. These tasks fall under the domain of *aerial physical interaction* (API), where the aerial robot must maintain stable flight while exerting controlled forces on external objects or surfaces.

Performing physical interaction with aerial robots introduces significant challenges. When a UAV establishes contact with the environment, the resulting interaction forces can destabilize the system due to the strong coupling between vehicle dynamics and contact forces. Environmental uncertainties such as unknown surface properties, friction variability, external disturbances, and sensor noise further complicate the control problem. As a result, designing controllers that ensure both flight stability and reliable interaction remains a major research challenge.

1.1 Classical Control Approaches for Aerial Robots

Classical control strategies for aerial vehicles typically rely on accurate dynamic models of the system. Techniques such as proportional–integral–derivative (PID) control, sliding mode control, and model predictive control have been widely used to stabilize quadrotor platforms and perform trajectory tracking tasks.

While these methods have proven effective for conventional flight operations, they often rely on precise knowledge of system parameters and environmental conditions. In real-world environments, however, disturbances such as wind, payload variations, and uncertain contact dynamics make accurate modeling difficult. Furthermore, aerial interaction tasks introduce complex nonlinear dynamics and strong coupling between motion and force control, which can limit the performance of traditional model-based controllers.

These limitations motivate the exploration of data-driven control approaches capable of adapting to uncertain and complex environments.

1.2 Reinforcement Learning for Aerial Robot Control

Recent advances in machine learning have introduced new possibilities for robotic control. Among these approaches, reinforcement learning (RL) has emerged as a powerful framework for learning control policies through interaction with the environment rather than relying solely on analytical models.

In reinforcement learning, an agent learns to perform a task by observing the state of the system, taking actions, and receiving rewards that guide the learning process. Over time, the agent optimizes its policy to maximize cumulative reward, thereby discovering effective control strategies.

Reinforcement learning has shown promising results in aerial robotics. Early work demonstrated that neural network policies could learn to stabilize quadrotor flight directly from system observations [1]. More recent studies have explored end-to-end reinforcement learning approaches capable of learning aggressive flight maneuvers and agile control behaviors [2, 3]. Advances in training efficiency have further enabled the rapid learning of aerial control policies, with recent methods demonstrating the ability to train quadrotor controllers within seconds in simulation [4].

Other research efforts have focused on improving the robustness and generalization capability of learned controllers. For example, learning-based quadrotor controllers capable of adapting to a wide range of system parameters and disturbances have been proposed [5]. Domain randomization techniques have also been used to train policies that generalize across different aerial platforms and environmental conditions [6].

Despite these advances, most reinforcement learning research in aerial robotics has focused on conventional quadrotor platforms operating in free-flight conditions.

1.3 Learning Control for Fully Actuated Aerial Robots

Recent developments in aerial hardware have introduced fully actuated aerial platforms capable of generating forces and torques independently across multiple degrees of freedom.

These systems include omnidirectional drones and overactuated aerial vehicles with tilted rotors.

Compared to traditional quadrotors, fully actuated aerial robots provide increased control authority and improved disturbance rejection. These capabilities are particularly advantageous for aerial interaction tasks, where maintaining a stable orientation while applying forces to external surfaces is often required.

However, the increased control dimensionality and actuator redundancy of these systems significantly complicate controller design. Reinforcement learning has recently been explored as a potential approach for controlling such platforms. For example, end-to-end reinforcement learning has been applied to omnidirectional aerial vehicles, demonstrating the feasibility of learning pose control policies for fully actuated drones [7].

Nevertheless, research on reinforcement learning control for fully actuated aerial robots remains relatively limited compared to the extensive literature on quadrotor systems.

1.4 Reinforcement Learning for Aerial Physical Interaction

Aerial physical interaction has emerged as an important research direction as aerial robots are increasingly required to perform manipulation and inspection tasks involving direct contact with the environment.

Several recent studies have explored reinforcement learning approaches for aerial interaction control. Learning-based variable impedance control strategies have been proposed for aerial sliding tasks, allowing drones to maintain stable contact with uneven surfaces while regulating interaction forces [8]. Similarly, reinforcement learning methods have been applied to variable admittance control for UAV interaction tasks, enabling adaptive behavior during contact with unknown environments [9].

More complex aerial manipulation tasks have also been investigated. For example, reinforcement learning has been used to enable aerial manipulators to perform tasks such as door opening, demonstrating the potential of learning-based approaches for aerial manipulation [10].

Despite these promising results, aerial physical interaction remains a relatively underexplored area. Most existing approaches rely on classical controllers for low-level stabilization while reinforcement learning is used primarily for higher-level decision making.

1.5 Research Gap and Motivation

The literature reviewed above highlights several important observations. First, the majority of reinforcement learning research in aerial robotics focuses on conventional

quadrotor platforms rather than fully actuated aerial robots. Second, many learning-based approaches rely on hierarchical control architectures where reinforcement learning is used only for high-level planning while traditional controllers handle low-level stabilization.

Furthermore, relatively few studies investigate reinforcement learning for aerial physical interaction tasks that involve sustained contact with the environment. Stable hovering performance is a fundamental prerequisite for such tasks, since the aerial robot must maintain precise position and orientation while experiencing external interaction forces.

Consequently, developing reinforcement learning policies capable of stabilizing fully actuated aerial robots represents an important step toward enabling aerial physical interaction capabilities.

1.6 Objectives and Contributions of the Thesis

The primary objective of this thesis is to investigate reinforcement learning control for a fully actuated aerial robot and to explore its potential for enabling aerial physical interaction tasks.

To achieve this objective, the research focuses on the development of reinforcement learning policies capable of stabilizing the aerial platform and supporting controlled interaction behaviors. The main contributions of this work can be summarized as follows:

- Development of a reinforcement learning framework for controlling a fully actuated aerial robot within physics-based simulation environments.
- Design of a progressive training strategy for learning stabilization behaviors, beginning with velocity damping and gradually extending to full hovering control.
- Implementation and evaluation of reinforcement learning policies for stable hovering and position tracking of the aerial platform.
- Investigation of contact modeling techniques for studying aerial interaction tasks in simulation.
- Evaluation of reinforcement learning performance using modern physics simulators such as MuJoCo [11] and Brax [12].

The development of a stable hovering controller represents a crucial step toward enabling more complex aerial manipulation and interaction behaviors.

1.7 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 reviews the state of the art in aerial robot control with a focus on reinforcement learning approaches and aerial interaction tasks.

Chapter 3 presents the dynamic modeling of the fully actuated aerial robot used in this work.

Chapter 4 describes the reinforcement learning formulation and control methodology.

Chapter 5 introduces the simulation environments and computational infrastructure used for training and evaluation.

Chapter 6 presents the development of the reinforcement learning hovering policy and analyzes its performance.

Chapter 7 investigates contact modeling and simulation experiments for aerial physical interaction.

Finally, Chapter 8 summarizes the main findings of the research and outlines directions for future work.

Chapter 2

Literature Review

Unmanned Aerial Vehicles (UAVs), particularly multirotor aerial robots, have experienced rapid growth in both academic research and industrial applications over the last decade. Their ability to perform vertical take-off and landing, hover in place, and maneuver in confined environments makes them suitable for tasks such as infrastructure inspection, environmental monitoring, disaster response, and aerial mapping. Traditionally, these systems have primarily operated in free-flight conditions where the robot performs sensing tasks without interacting physically with the surrounding environment.

However, many real-world applications require aerial robots to interact directly with their environment. Examples include contact-based inspection of structures, valve turning, door opening, surface sampling, and maintenance tasks in hazardous or inaccessible locations. These tasks fall under the domain of *Aerial Physical Interaction (API)*, where the robot must maintain stable flight while applying forces to external structures.

Aerial physical interaction introduces several fundamental challenges. When an aerial robot establishes contact with the environment, the resulting forces can destabilize the system due to the coupling between the robot's dynamics and the interaction forces. Maintaining stability under such disturbances requires robust and responsive control strategies. In addition, environmental uncertainties, such as unknown surface properties or external disturbances, further complicate controller design.

Recent advances in machine learning, particularly Reinforcement Learning (RL), have opened new opportunities for control design in complex robotic systems. RL allows agents to learn control policies through interaction with an environment rather than relying on precise analytical models. This paradigm is especially appealing for aerial robotics, where system dynamics are nonlinear, uncertain, and often difficult to model accurately.

Despite these advantages, most reinforcement learning research in aerial robotics has focused on traditional quadrotor platforms and tasks such as trajectory tracking, aggressive flight, and drone racing. In contrast, the application of RL to fully actuated aerial robots and aerial interaction tasks remains relatively unexplored. Furthermore, learning stable

hovering policies for complex aerial platforms is itself a challenging problem due to the high dimensionality and nonlinear dynamics of the system.

This chapter reviews the state of the art in aerial robot control with an emphasis on reinforcement learning methods. The literature is presented in chronological order from the most recent developments to earlier foundational work. The review highlights key advances in learning-based aerial control, discusses research on aerial physical interaction, and identifies gaps in the current literature that motivate the research presented in this thesis.

2.1 Recent Advances in Learning-Based Aerial Control

In recent years, reinforcement learning has gained significant attention as a promising approach for controlling aerial robots. One of the primary motivations for applying RL to aerial robotics is the ability to learn complex control policies without requiring precise system models.

Eschmann et al. [4] recently proposed a reinforcement learning framework capable of learning end-to-end quadrotor control in extremely short training times. Their approach employs an asymmetric actor-critic architecture combined with curriculum learning and an optimized simulation environment. The authors demonstrate that a quadrotor control policy can be trained in seconds and deployed directly to a real platform while maintaining competitive trajectory tracking performance. Their work highlights the potential of RL for low-level aerial control while also emphasizing challenges related to simulation fidelity and training stability.

Similarly, Zhang et al. [5] introduced a learning-based controller designed to operate across a wide range of quadcopter platforms with varying physical parameters. Their framework combines imitation learning and reinforcement learning to train controllers capable of adapting to disturbances such as payload variations, actuator failures, and wind disturbances. Experimental results demonstrate that the controller can adapt to quadrotors with significantly different physical properties, suggesting that learning-based methods can generalize across platforms more effectively than traditional controllers.

Another line of research has focused on improving the generalization capability of learned controllers. Ferede et al. [6] proposed a neural network controller capable of operating across different drone platforms using domain randomization techniques during training. Their results demonstrate that a single controller can successfully control multiple quadrotor configurations in drone racing scenarios. However, this work also highlights the trade-off between robustness and optimality introduced by domain randomization.

Although these studies demonstrate significant progress in RL-based aerial control, they primarily address conventional quadrotor platforms operating in free-flight conditions. The control architectures in these works typically assume underactuated dynamics and rely on hierarchical control stacks in which high-level policies generate commands that are executed by low-level controllers.

2.2 End-to-End Reinforcement Learning for Aerial Systems

A growing body of research has investigated end-to-end reinforcement learning approaches that directly map system observations to actuator commands. These approaches aim to eliminate traditional control pipelines and instead learn complete control policies through interaction with the environment.

Ferede et al. [3] proposed an end-to-end reinforcement learning method for achieving time-optimal quadcopter flight. Their framework trains neural policies that directly output motor commands while compensating for modeling errors through adaptive residual dynamics models. The results show that the learned controller can achieve faster flight trajectories compared to traditional control approaches.

Kaufmann et al. [2] conducted one of the first benchmark comparisons of learned control policies for agile quadrotor flight. Their study evaluated multiple control abstractions, including policies that output velocity commands, body rate commands, and direct motor thrusts. Their results indicate that policies generating intermediate-level commands, such as collective thrust and body rates, provide better robustness when transferring policies from simulation to real-world platforms.

These findings illustrate an important trade-off in reinforcement learning control for aerial robots. Lower-level control representations offer greater control authority and potentially higher performance but are more difficult to train and transfer to real systems due to the simulation-to-reality gap. Higher-level abstractions improve transferability but restrict the expressiveness of the learned policy.

2.3 Learning Control for Fully Actuated and Omnidirectional Aerial Robots

While most reinforcement learning research in aerial robotics focuses on standard quadrotors, recent developments in aerial hardware have introduced fully actuated aerial platforms capable of generating forces and torques independently in all six degrees of freedom.

These platforms include omnidirectional drones, tilt-rotor systems, and other overactuated aerial vehicles. Unlike conventional quadrotors, fully actuated drones can apply forces in arbitrary directions without requiring large attitude changes. This capability is particularly beneficial for aerial physical interaction tasks where maintaining a fixed orientation while exerting forces on a surface is desirable.

Cuniato et al. [7] investigated reinforcement learning control for an omnidirectional micro aerial vehicle (OMAV). Their work demonstrated that RL can learn pose control policies capable of exploiting actuator redundancy in tilt-rotor systems. The results show improved disturbance rejection compared to classical model-based controllers.

However, fully actuated aerial robots introduce additional complexity in controller design. The presence of additional actuators increases the dimensionality of the control problem, while the coupling between propeller dynamics and tilting mechanisms introduces nonlinearities that are difficult to model accurately. As a result, learning stable control policies for these systems remains an open challenge.

2.4 Reinforcement Learning for Aerial Physical Interaction

Aerial physical interaction has emerged as an important research area as aerial robots are increasingly required to perform manipulation and inspection tasks that involve contact with the environment.

Zhang et al. [8] proposed a learning-based variable impedance control strategy for aerial sliding tasks. Their approach uses tactile and proprioceptive sensing to adjust controller gains in response to environmental changes. The system is trained in simulation and successfully deployed on a real omnidirectional aerial robot performing surface sliding tasks.

Feng et al. [9] introduced a reinforcement learning approach for variable admittance interaction control in UAVs. Their framework allows the robot to automatically adapt interaction parameters based on environmental feedback, enabling safer and more stable interaction with unknown environments.

More complex interaction behaviors have also been explored. Cuniato et al. [10] demonstrated that reinforcement learning can be used to learn aerial manipulation policies for tasks such as opening doors with an aerial manipulator. Their approach was trained in simulation and successfully transferred to real-world experiments, showing improved robustness compared to model predictive control methods.

Despite these promising developments, aerial physical interaction research remains limited. Most existing studies rely on carefully tuned low-level controllers to maintain stable flight while reinforcement learning is used primarily for higher-level decision-making.

This highlights the importance of developing robust low-level stabilization policies before attempting complex interaction tasks.

2.5 Early Reinforcement Learning Approaches for UAV Control

One of the earliest demonstrations of reinforcement learning for quadrotor control was presented by Hwangbo et al. [1]. Their work showed that neural network policies trained in simulation could directly map system states to rotor thrust commands and stabilize a quadrotor in real-world experiments.

The authors demonstrated that the learned controller could recover from aggressive disturbances such as being thrown upside down. This work established the feasibility of reinforcement learning for low-level aerial control and inspired subsequent research exploring data-driven control strategies for UAVs.

However, these early studies also revealed several challenges associated with reinforcement learning control, including high sample complexity, sensitivity to reward design, and difficulties in transferring policies from simulation to real systems.

Reference	Platform	Control Level	Task Description
Eschmann et al. (2024)	Quadrotor	End-to-End RL	Learning low-level motor commands directly from system states for hovering and trajectory tracking.
Zhang et al. (2024)	Quadrotor	RL + Imitation	Adaptive controller capable of handling variations in mass, propeller properties, and disturbances.
Ferede et al. (2025)	Quadrotor	End-to-End RL	Generalized neural controller trained using domain randomization to operate across multiple drone platforms.
Ferede et al. (2023)	Quadrotor	End-to-End RL	Learning time-optimal aggressive flight policies using reinforcement learning.
Kaufmann et al. (2022)	Quadrotor	RL (Body Rate Control)	Benchmark comparison of learned control policies for agile flight.
Cuniato et al. (2024)	OMAV (Fully Actuated)	RL Pose Control	End-to-end reinforcement learning controller for omnidirectional aerial vehicles.
Zhang et al. (2022)	OMAV	RL + Impedance Control	Learning variable impedance control for aerial surface interaction tasks.
Cuniato et al. (2023)	OMAV Manipulator	RL	Learning aerial manipulation behaviour for door opening tasks.
Feng et al. (2023)	Quadrotor	RL Interaction Control	Variable admittance control using reinforcement learning for force interaction tasks.
Hwangbo et al. (2017)	Quadrotor	End-to-End RL	Early demonstration of reinforcement learning for quadrotor stabilization.

Table 2.1: Comparison of reinforcement learning approaches for aerial robot control.

2.6 Comparison of Learning-Based Aerial Control Approaches

To better understand the current state of research in reinforcement learning for aerial robotics, Table 2.1 summarizes the most relevant works discussed in this chapter. The comparison highlights the type of aerial platform, the level of control learned by the policy, and the primary task addressed in each study.

2.7 Evolution of Learning-Based Aerial Control

The development of aerial robotics control strategies has progressed through several stages over the past decades. Initially, research focused on classical model-based control methods such as PID control, sliding mode control, and model predictive control. These approaches rely on accurate system modeling and extensive manual tuning.

With the emergence of machine learning techniques, reinforcement learning began to be explored as an alternative control paradigm capable of learning control policies directly from interaction data. Early studies demonstrated RL-based stabilization for quadrotor systems, while more recent work has explored aggressive flight, drone racing, and trajectory optimization.

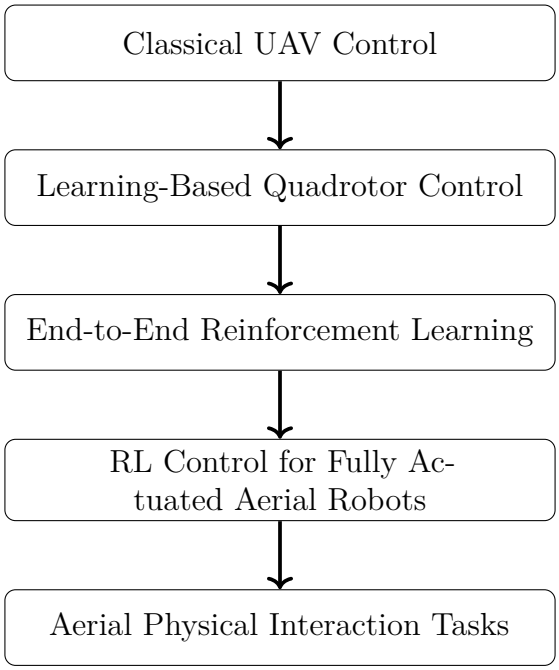

Figure 2.1: Evolution of aerial robot control research leading toward aerial physical interaction.

More recently, research has begun to investigate aerial physical interaction, where aerial robots must maintain stable flight while applying forces to the environment. This domain

introduces new challenges related to contact dynamics, environmental uncertainty, and disturbance rejection.

As illustrated in Figure 2.1, research on reinforcement learning for fully actuated aerial robots remains relatively limited compared to the extensive literature on quadrotor control.

2.8 Research Gaps and Motivation

The literature reviewed above highlights several important observations.

First, the majority of reinforcement learning research in aerial robotics focuses on conventional quadrotor platforms rather than fully actuated aerial robots. These systems have relatively simple dynamics and well-understood control architectures.

Second, many RL-based control approaches rely on hierarchical control structures where reinforcement learning is used only for high-level decision making while traditional controllers handle low-level stabilization.

Third, relatively few studies investigate reinforcement learning for low-level stabilization of complex aerial platforms such as omnidirectional drones.

Finally, aerial physical interaction tasks require extremely stable hovering performance since the robot must maintain precise positioning while experiencing unpredictable contact forces.

Consequently, the development of a stable hovering policy for a fully actuated aerial robot represents a fundamental prerequisite for enabling aerial physical interaction tasks. The absence of extensive research in this area further emphasizes the significance of this problem.

2.9 Positioning of the Thesis Contribution

Based on the literature reviewed in this chapter, it is evident that reinforcement learning has demonstrated strong potential for controlling aerial robots, particularly conventional quadrotor platforms. However, several limitations remain in the current body of research.

First, the majority of reinforcement learning studies focus on standard quadrotor systems, whose control structure and dynamics are well understood. These platforms are underactuated and typically rely on hierarchical control architectures consisting of position, attitude, and motor control loops.

Second, many reinforcement learning approaches are applied only to higher-level control tasks such as trajectory planning, waypoint navigation, or racing strategies. In these cases, classical controllers remain responsible for the fundamental task of stabilizing the aerial platform.

Third, research addressing reinforcement learning control for fully actuated aerial robots remains sparse. Fully actuated aerial platforms introduce additional challenges due to

their increased number of actuators, higher control dimensionality, and more complex system dynamics.

These challenges become particularly significant in the context of aerial physical interaction. In such scenarios, the aerial robot must maintain extremely stable hovering performance while experiencing unpredictable external forces resulting from contact with the environment.

Consequently, developing a stable hovering control policy for a fully actuated aerial robot represents a critical prerequisite before attempting more complex interaction tasks. Unlike trajectory tracking or racing scenarios, hovering stabilization requires continuous disturbance rejection and precise control authority across all degrees of freedom.

For this reason, a significant portion of this thesis focuses on the development and implementation of a reinforcement learning-based hovering policy for a fully actuated aerial robot. Although this task may appear simple at first glance, the complexity of the platform dynamics and the limited availability of prior research in this area make it a challenging and meaningful research problem.

The successful development of a stable hovering policy establishes the necessary foundation for future work on aerial physical interaction tasks, which represent the long-term objective of this research.

2.10 Summary

This chapter reviewed the state of the art in aerial robot control with a focus on reinforcement learning approaches. While reinforcement learning has demonstrated promising results in quadrotor control and aggressive flight scenarios, its application to fully actuated aerial robots and aerial interaction tasks remains limited.

Existing work highlights the importance of robust low-level control policies as a foundation for higher-level behaviors. In particular, stable hovering control is a critical requirement for enabling aerial physical interaction.

The research presented in this thesis focuses on the development of reinforcement learning policies for stabilizing a fully actuated aerial robot. This work represents an important step toward enabling aerial robots to perform complex physical interaction tasks in real-world environments.

Transition to System Modeling

The literature reviewed in this chapter highlights significant progress in the application of reinforcement learning for aerial robot control. However, most existing approaches focus on conventional quadrotor platforms and rely on hierarchical control architectures in which reinforcement learning is applied primarily at higher levels of the control stack.

In contrast, the development of reinforcement learning policies for fully actuated aerial robots remains relatively unexplored. These platforms introduce additional challenges due to their increased control dimensionality, actuator redundancy, and more complex system dynamics. Furthermore, aerial physical interaction tasks require highly stable hovering capabilities to safely maintain contact with external structures.

Consequently, a detailed understanding of the system dynamics and control architecture of the aerial platform is essential before designing and training reinforcement learning policies. The following chapter therefore introduces the aerial robot considered in this thesis and presents the mathematical modeling of its dynamics. This modeling framework provides the foundation for the reinforcement learning control strategies developed in the subsequent chapters.

Chapter 3

Aerial Robot Dynamical Modeling

This section presents the dynamic model of the fully–actuated aerial robot (AR) considered in this work. The formulation of the model is required for both simulation and control design of RL environments, particularly for studying interaction tasks between the aerial platform and the environment. The considered system is of a hexarotor platform equipped with fixedly tilted rotors and an end–effector mounted on the structure as in Fig. 3.1.

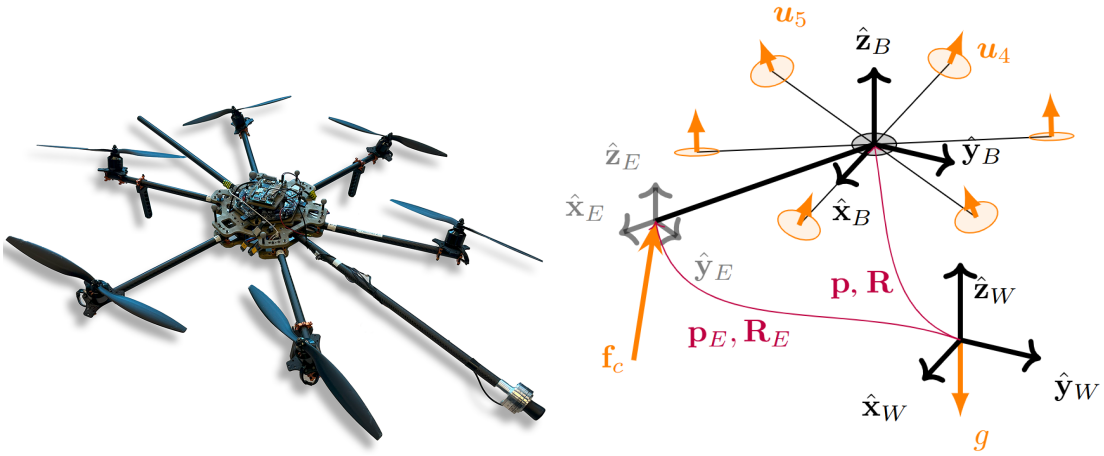

Figure 3.1: (a) Fully–actuated hexarotor platform with fixedly tilted rotors and an interaction tool equipped with a force/torque sensor. (b) Schematic representation of the robot and its reference frames.

3.1 Reference Frames

To describe the robot dynamics, several coordinate frames are introduced.

- The inertial world frame is denoted by

$$\mathcal{F}_W = O_W\{\hat{x}_W, \hat{y}_W, \hat{z}_W\} \tag{3.1}$$

where O_W is the origin and $\hat{x}_W, \hat{y}_W, \hat{z}_W$ are the unit basis vectors.

- The body frame is defined as

$$\mathcal{F}_B = O_B\{\hat{x}_B, \hat{y}_B, \hat{z}_B\} \tag{3.2}$$

which is attached to the geometric center of the aerial robot. The axes of $\mathcal{F}_B$ are aligned with the principal inertia axes of the rigid body, such that the inertia matrix

$$J \in \mathbb{R}^{3\times3}_{>0} \tag{3.3}$$

is diagonal.

- The end–effector frame is defined as

$$\mathcal{F}_E = O_E\{\hat{x}_E, \hat{y}_E, \hat{z}_E\} \tag{3.4}$$

which is rigidly attached to the interaction tool mounted on the robot.

- Each actuator i is associated with a frame

$$\mathcal{F}_{A_i} = O_{A_i}\{\hat{x}_{A_i}, \hat{y}_{A_i}, \hat{z}_{A_i}\}, \quad i \in \{1, \dots, 6\} \tag{3.5}$$

where O_{A_i} coincides with the thrust generation point and $\hat{z}_{A_i}$ is aligned with the thrust direction of the i-th rotor.

3.2 State Representation

The translational motion of the aerial robot is described by the position of the body frame with respect to the inertial frame. Let

$$p, \dot{p}, \ddot{p} \in \mathbb{R}^3 \tag{3.6}$$

denote the position, velocity, and acceleration of $\mathcal{F}_B$ with respect to $\mathcal{F}_W$, expressed in the inertial frame.

The position of the end–effector with respect to the body frame is defined as

$$p_E^B \in \mathbb{R}^3. \tag{3.7}$$

The orientation of the body frame relative to the world frame is represented by the rotation matrix

$$R_B \in SO(3) \tag{3.8}$$

while the angular velocity and angular acceleration of the $\mathcal{F}_B$ with respect to $\mathcal{F}_W$ are given by

$$\omega, \dot{\omega} \in \mathbb{R}^3 \tag{3.9}$$

expressed in the body frame.

3.3 Equations of Motion

For an aerial robot with mass $m \in \mathbb{R}_{>0}$ and inertia matrix J, the rigid–body dynamics can be expressed using the Newton–Euler equations.

Translational Dynamics

The translational dynamics are given by

$$mI_3\ddot{p} = -mg\hat{z}_W + R_B G_f u + f_c + f_e \tag{3.10}$$

where

- $I_3 \in \mathbb{R}^{3\times3}$ is the identity matrix,
- g is the gravitational acceleration,
- $u \in \mathbb{R}^6$ represents the rotor thrust inputs,
- $G_f \in \mathbb{R}^{3\times6}$ is the force allocation matrix,
- $f_c \in \mathbb{R}^3$ denotes contact forces applied at the end–effector,
- $f_e \in \mathbb{R}^3$ represents external disturbance forces.

Rotational Dynamics

The rotational motion of the robot is described by

$$J\dot{\omega} = -\omega \times J\omega + G_\tau u + \tau_c + \tau_e \tag{3.11}$$

where

- $G_\tau \in \mathbb{R}^{3\times6}$ is the moment allocation matrix,
- $\tau_c \in \mathbb{R}^3$ represents torques generated by contact interaction,
- $\tau_e \in \mathbb{R}^3$ denotes external disturbance torques.

The kinematic relationship governing the orientation evolution is

$$\dot{R}_B = R_B[\omega]_\times \tag{3.12}$$

where $[\cdot]_\times$ denotes the skew–symmetric matrix associated with a vector in $\mathbb{R}^3$.

3.4 Interaction Forces and Torques

The aerial robot is assumed to interact with the environment through the end–effector. The contact force f_c applied at the end–effector generates a moment about the body–frame origin due to the offset between the contact point and the body center. This contact–induced torque is given by

$$\tau_c = p_E^B \times f_c. \tag{3.13}$$

In addition to contact forces, the system may be affected by external disturbances represented by f_e and τ_e, expressed in the world and body frames respectively.

3.5 Control Allocation Matrix

The mapping between rotor thrusts and the resulting forces and torques is described by the allocation matrix

$$G \in \mathbb{R}^{6\times 6}. \tag{3.14}$$

The j-th column of G is defined as

$$G(:,j) = \begin{bmatrix} R_B R_{A_j}^B \hat{z}_{A_j} \\ \left([p_{A_j}^B]_\times + k_j c_{d_j} I_3\right) R_{A_j}^B \hat{z}_{A_j} \end{bmatrix} \tag{3.15}$$

where

- $p_{A_j}^B$ is the position of the j-th actuator frame with respect to the body frame,
- $R_{A_j}^B$ represents the orientation of the actuator frame relative to the body frame,
- c_{d_j} is the drag coefficient of rotor j,
- $k_j \in \{-1, 1\}$ represents the rotor spin direction (counter–clockwise or clockwise).

3.6 Physical Parameters of the Aerial Robot

The physical parameters of the aerial platform used in this work are summarized in Table 3.1.

Table 3.1: Aerial Robot Physical Parameters

Parameter	Symbol	Value
Mass	m	2.0 kg
Inertia	J	diag(0.062, 0.465, 0.472) kg·m^2
Arm length	l	0.39 m
Rotor tilt angle	α	$\pm 20^\circ$
Maximum thrust	$\bar{u}$	14 N
Maximum thrust rate	$\dot{\bar{u}}$	25 N/s
Minimum thrust	$\underline{u}$	0 N
Minimum thrust rate	$\dot{\underline{u}}$	-15 N/s
Rotor thrust coefficient	c_f	11.75×10^{-4} N/Hz^2
Rotor drag coefficient	c_d	2.388×10^{-5} Nm/Hz^2
Motor time constant	d	0.005

3.7 Actuator Dynamics

The rotor dynamics are modeled as a first–order system to account for actuator delays. The thrust dynamics are given by

$$\dot{u} = \frac{1}{d}(u^{ref} - u) \tag{3.16}$$

where

- $u^{ref} \in \mathbb{R}^6$ represents the commanded rotor thrust,
- $d \in \mathbb{R}_{>0}$ is the motor time constant.

The thrust inputs and their derivatives are subject to actuator constraints

$$\underline{u} \leq u \leq \bar{u} \tag{3.17}$$

$$\dot{\underline{u}} \leq \dot{u} \leq \dot{\bar{u}} \tag{3.18}$$

which reflect the physical limitations of the propulsion system.

Modeling actuator dynamics during training is crucial to ensure both the stability and transferability of learned control policies. Moreover, respecting actuator constraints prevents instability that may arise from unrealistic control commands, as discussed in previous studies.

Chapter 4

Methodology

This chapter presents the methodology planned to be used to learn a force–motion control policy for a fully actuated aerial robot interacting with a surface. The problem is formulated as a reinforcement learning task within the framework of a Markov Decision Process (MDP). The design includes the formulation of the control policy, definition of the observation and action spaces, reward design, termination conditions, reference command generation, and domain randomization strategies used during training. The overall training pipeline is illustrated in Fig. 4.1.

4.1 Problem Formulation

The learning problem is modeled as a Markov Decision Process (MDP) defined as

$$\mathcal{M} = (S, A, P, r, \gamma) \tag{4.1}$$

where

- S represents the state space
- A denotes the action space
- $P(s_{t+1}|s_t, a_t)$ represents the stochastic transition dynamics
- $r(s_t, a_t)$ is the reward function
- $\gamma \in [0, 1]$ is the discount factor

The objective is to learn a policy

$$\pi_\theta : S \to A \tag{4.2}$$

parameterized by θ, that maximizes the expected discounted return

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] \tag{4.3}$$

The system trajectory evolves according to the stochastic dynamics

$$s_{t+1} \sim P(s_{t+1}|s_t, a_t), \quad a_t \sim \pi_\theta(a_t|s_t) \tag{4.4}$$

The learning objective is to obtain a control policy that produces stable aerial contact interactions while maintaining precise force and position regulation.

4.2 Action Space

The policy outputs normalized thrust commands defined as

$$a_t \in \mathbb{R}^6 \tag{4.5}$$

These actions represent the normalized thrust signals sent to the robot actuators. Each component of the action vector is bounded within the actuator operating limits. The normalized actions are scaled to produce the actual thrust commands

$$u_t \in \mathbb{R}^6 \tag{4.6}$$

which represent the physical thrust applied by the motors.

To simulate realistic actuator dynamics, a motor model is applied to transform the commanded actions into realizable thrust values.

4.2.1 Thrust Allocation

The thrust commands are converted into body-frame wrench vectors using an input allocation matrix

$$G \in \mathbb{R}^{6\times 6} \tag{4.7}$$

This matrix maps the individual motor thrusts to the resulting force and torque applied to the robot body.

$$w_t = G u_t \tag{4.8}$$

where

- $w_t \in \mathbb{R}^6$ represents the wrench applied to the robot
- the wrench includes both translational forces and rotational torques

This mapping captures the mechanical relationship between the propeller thrusts and the resulting motion of the aerial robot.

4.3 Observation Space

At each timestep the agent receives an observation vector

$$o_t \in \mathbb{R}^d \tag{4.9}$$

which includes both proprioceptive and exteroceptive measurements necessary for contact-aware aerial control.

The observation vector is defined as

$$o_t = \left[p^T, vec(R_B)^T, v^T, \omega^T, f_c^T, p_E^T, vec(R_E)^T, p_E^{ref}, f^{ref}, a_{prev}, u_{prev}\right]^T \tag{4.10}$$

where

- p is the robot position in the world frame
- R_B is the orientation of the robot body
- v is the linear velocity
- ω is the angular velocity
- f_c represents the measured contact force
- p_E is the end-effector position
- R_E represents the end-effector orientation
- p_E^{ref} is the desired end-effector position
- f^{ref} is the desired normal contact force
- a_{prev} represents the previous action
- u_{prev} represents previously realized thrusts

All matrix quantities are flattened using column-wise vectorization before being concatenated into the observation vector.

Including previous actions in the observation helps the policy maintain temporal smoothness and reduces high-frequency control oscillations.

4.4 Reward Function

To guide the learning process, a composite reward function is defined. The reward is designed to encourage accurate positioning, force tracking, smooth control actions, and stable robot orientation.

The overall reward at timestep t is defined as

$$r = r_{pos} + r_{pos\_fine} + r_{force} + r_{act} + r_{flat} + r_{term} \tag{4.11}$$

Each term in the reward function is weighted by a scaling coefficient.

4.4.1 Position Tracking Reward

The coarse position reward encourages the end-effector to move toward the desired reference position.

$$r_{pos} = \lambda_1 \left(1 - \tanh\left(\frac{\|p_E - p_E^{ref}\|_2}{\sigma}\right)\right) \tag{4.12}$$

4.4.2 Fine Position Reward

A second reward term provides fine-grained shaping when the end-effector approaches the target.

$$r_{pos\_fine} = \lambda_1' \left(1 - \tanh\left(\frac{\|p_E - p_E^{ref}\|_2}{\sigma'}\right)\right) \tag{4.13}$$

This term ensures precise positioning close to the contact point.

4.4.3 Force Tracking Reward

The force reward penalizes deviations from the desired contact force.

$$r_{force} = \lambda_2 \|f_{c,z} - f^{ref}\|^2 \tag{4.14}$$

where $f_{c,z}$ is the measured contact force along the surface normal.

4.4.4 Action Smoothness Reward

To reduce abrupt control signals, a smoothness penalty is applied to the change in actions.

$$r_{act} = \lambda_3 \|a - a_{prev}\|^2 \tag{4.15}$$

4.4.5 Orientation Alignment Reward

To maintain stable contact with the surface, the robot should remain approximately level during interaction. Therefore, an orientation alignment reward is introduced to penalize deviations from the desired orientation.

The orientation alignment reward is defined as

$$r_{flat} = \lambda_4 \left\| \frac{1}{2} \left(R_d^T R_B - R_B^T R_d \right)^V \right\|^2 \tag{4.16}$$

where R_B denotes the current body rotation matrix of the robot and R_d represents the desired orientation. The operator $(\cdot)^V$ converts the skew-symmetric matrix into its corresponding vector form.

In this work, the desired orientation is chosen as $R_d = I_3$ which corresponds to a flat orientation of the drone with respect to the world frame. This reward term penalizes orientation deviations and encourages the robot to maintain a stable posture during contact interaction.

4.4.6 Termination Penalty

A termination penalty is applied when an episode ends prematurely due to unsafe conditions.

$$r_{term} = \lambda_5 \cdot \mathbb{1}_{terminated} \tag{4.17}$$

4.5 Episode Termination Conditions

Training episodes terminate when unsafe or invalid robot states occur. The termination conditions include:

- Collision of the robot body with the surface
- Loss of sustained contact at the end-effector
- Robot orientation exceeding 60° from upright
- End-effector deviating more than 2 meters from the reference position

These constraints ensure that the learning process remains within safe operational limits.

4.6 Reference Command Generation

At every timestep the agent receives reference commands that specify the desired interaction with the surface.

The reference signals include:

- Desired contact position p_E^{ref}
- Desired normal contact force f^{ref}

Depending on the task configuration, the reference position can

- remain constant
- vary linearly over time
- follow predefined trajectories

These references guide the policy toward achieving controlled contact with the surface.

4.7 Training Framework

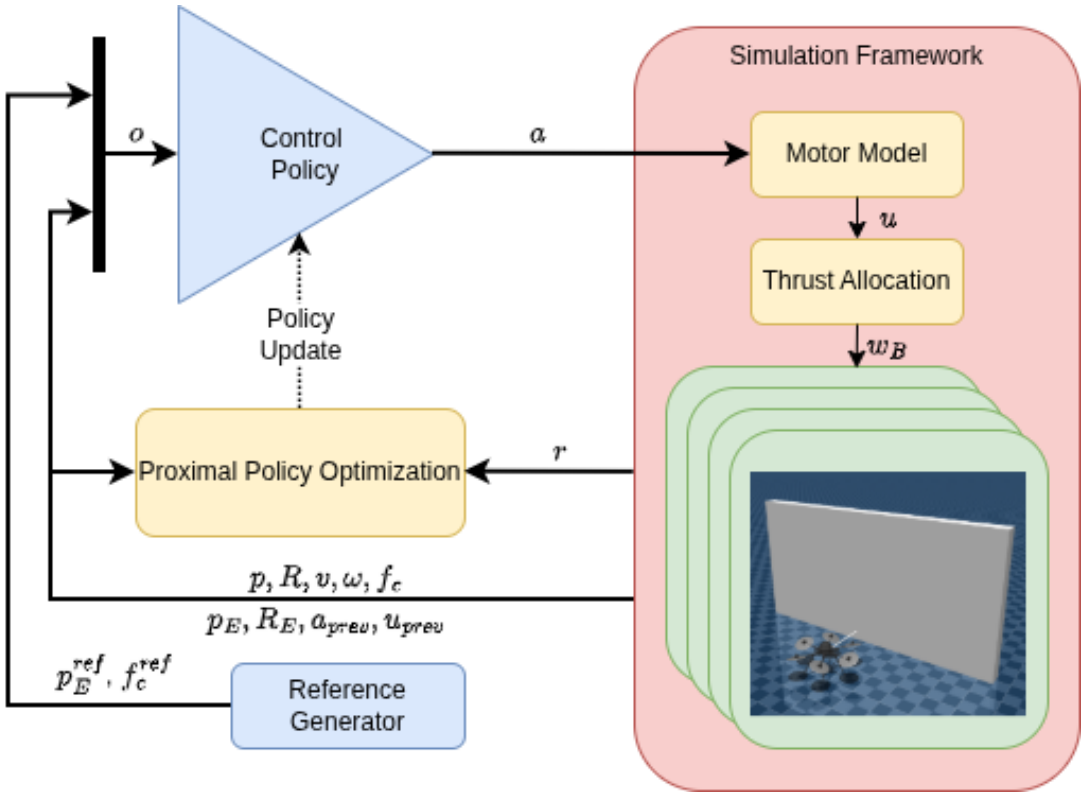

Figure 4.1: Training framework for learning aerial force–motion interaction policies.

The overall training pipeline is illustrated in Fig. 4.1. The control policy receives observations from the environment and outputs thrust commands. The motor model and thrust allocation module translate these commands into forces applied to the robot in simulation.

The policy is optimized using the Proximal Policy Optimization (PPO) algorithm, which updates the policy parameters using collected trajectories and computed rewards.

Through iterative training, the policy learns to generate stable control commands that enable reliable aerial interaction with surfaces.

Chapter 5

Training and Testing Setup

This chapter describes the simulation environments, training methodology, and computational infrastructure used for developing and evaluating the reinforcement learning (RL) agents in this work. Reinforcement learning experiments often require a large number of interactions between the agent and the environment, making efficient simulation frameworks essential. To facilitate scalable experimentation and rapid prototyping, physics-based simulation environments were used instead of real-world hardware.

Two simulation frameworks were used in this work: MuJoCo and Brax. These environments provide realistic physics simulation for articulated systems and are widely used in reinforcement learning research for control tasks. The use of simulation allows the agent to interact with the environment millions of times during training without the risks and limitations associated with physical systems.

Additionally, the training process was conducted across multiple hardware platforms due to varying resource availability during the research period. The following sections describe the simulation environments and hardware configurations used for the training and evaluation of the reinforcement learning models.

5.1 Simulation Environments

Physics simulation plays a crucial role in reinforcement learning research, particularly in robotics and control applications. Accurate simulation allows researchers to model complex dynamics while enabling rapid experimentation and reproducibility. In this work, two different physics simulation frameworks were used: MuJoCo and Brax. Each framework offers unique advantages in terms of computational efficiency and scalability.

5.1.1 MuJoCo Physics Simulator

MuJoCo (Multi-Joint dynamics with Contact) is a physics engine designed for high-performance simulation of articulated rigid-body systems [11]. It is widely used in

reinforcement learning research due to its accurate modeling of physical interactions and efficient computation of multi-body dynamics.

MuJoCo uses generalized coordinates and advanced numerical optimization methods to compute system dynamics efficiently. The simulator is capable of modeling complex robotic systems with multiple joints, constraints, and contact interactions. These features make it particularly suitable for continuous control tasks commonly used in reinforcement learning benchmarks.

One of the key advantages of MuJoCo is its ability to perform efficient simulations using standard CPU resources. Unlike many modern simulation frameworks that rely heavily on GPU acceleration, MuJoCo is optimized for CPU-based computation. This characteristic makes it especially useful for research environments where high-end GPU hardware may not be available.

During the initial stages of this research, training experiments were primarily conducted using CPU-based simulations in MuJoCo. This approach was necessary because the available GPU hardware had limited memory capacity and was insufficient for large-scale reinforcement learning workloads. The CPU-based training approach enabled stable experimentation and allowed the development of reinforcement learning pipelines without requiring high-end GPU resources.

The CPU efficiency of MuJoCo offers several practical advantages:

- **Hardware Accessibility:** Training can be performed on standard desktop computers without requiring specialized GPU hardware.
- **Stable Simulation Performance:** CPU-based simulations often provide stable and deterministic execution, which is beneficial for debugging and algorithm development.
- **Memory Efficiency:** CPU simulations typically consume less GPU memory, allowing experimentation even on systems with limited graphics resources.
- **Ease of Development:** The CPU-based workflow simplifies the integration of reinforcement learning algorithms and simulation environments.

Due to these advantages, MuJoCo served as the primary simulation environment during the early stages of the research. It enabled rapid development and testing of reinforcement learning algorithms while maintaining reliable simulation accuracy.

In addition to CPU-based execution, MuJoCo provides detailed state observations including joint positions, velocities, contact forces, and other physical parameters. These observations form the state space used by the reinforcement learning agent to make control decisions. The reward function is computed based on task-specific objectives such as stability, movement efficiency, or trajectory tracking.

5.1.2 Brax Physics Engine

Brax is a modern physics simulation engine developed by Google Research that is designed for reinforcement learning and large-scale physics simulation [12]. The framework is built using the JAX numerical computing library, allowing it to execute efficiently on multiple hardware backends including CPUs, GPUs, and TPUs.

Unlike many traditional physics engines, Brax is designed with parallelism and differentiability in mind. By leveraging JAX's just-in-time (JIT) compilation and vectorized computation capabilities, Brax can simulate multiple environments simultaneously, significantly improving training efficiency for reinforcement learning algorithms.

One of the key advantages of Brax is its hardware flexibility. Since the simulator is implemented within the JAX ecosystem, the same simulation code can be executed on different hardware platforms without modification. This allows experiments to be conducted on CPU-based systems during early development stages and later scaled to GPU or accelerator-based systems when additional computational resources are available.

In this work, Brax was used both on CPU and GPU platforms depending on the available hardware. During early experimentation and debugging stages, Brax simulations were executed on CPU-based systems. This allowed the reinforcement learning pipeline to be tested and validated without requiring specialized GPU resources.

When more powerful hardware became available, Brax benefited from GPU acceleration, enabling multiple simulation environments to run in parallel. This significantly increased training throughput by allowing the agent to collect experience from many environments simultaneously.

The main advantages of Brax include:

- **Hardware Flexibility:** The simulator can run on CPUs, GPUs, or TPUs through the JAX framework.
- **Parallel Environment Simulation:** Multiple environments can be simulated simultaneously using vectorized operations.
- **Integration with Machine Learning Frameworks:** Brax integrates naturally with modern deep learning workflows through JAX.
- **Accelerated Training:** Parallel simulation allows faster policy optimization compared to single-environment simulation.

Due to these properties, Brax provided a flexible simulation platform that could adapt to different hardware configurations used throughout this research.

5.2 Hardware Platforms

The reinforcement learning experiments conducted in this research required significant computational resources due to the large number of environment interactions required during training. However, the availability of computational resources varied throughout the project timeline. As a result, experiments were performed on three different hardware platforms depending on availability.

5.2.1 Local Development System

The initial stages of the project were conducted on a personal development computer. This system was primarily used for environment setup, algorithm development, debugging, and preliminary training experiments.

The specifications of the local development machine are as follows:

- Processor: Intel Core i5 (8th Generation)
- Memory: 32 GB RAM
- Graphics Processing Unit: NVIDIA GeForce GTX 1050 with 4 GB VRAM

Due to the limited GPU memory available on the GTX 1050, large-scale GPU training was not feasible on this system. Therefore, most early experiments relied on CPU-based training using the MuJoCo simulator. Despite the hardware limitations, this setup was sufficient for testing reinforcement learning algorithms and validating experimental configurations.

5.2.2 NVIDIA Compute Server

For computationally intensive experiments, training was performed on a GPU-enabled compute server. The system was a Dell PowerEdge R7525 equipped with two AMD EPYC 7313 processors (16 cores each) and 256 GB of DDR4 memory.

Deep learning workloads were accelerated using two NVIDIA A30 GPUs based on the Ampere architecture, each providing 24 GB of GPU memory. The server also included high-speed enterprise SSD storage configured with a hardware RAID controller.

Compared to the local development machine, this compute infrastructure enabled significantly faster training and allowed longer training runs and larger batch sizes, improving the efficiency of the reinforcement learning experiments.

5.2.3 Workstation System

In addition to the development machine and server resources, some experiments were also conducted on a dedicated workstation. This system provided improved GPU capabilities compared to the initial development setup.

The workstation configuration consisted of:

- Processor: AMD Ryzen 5
- Memory: 8 GB DDR4 RAM
- Graphics Processing Unit: NVIDIA RTX 3060

The RTX 3060 GPU significantly improved training performance compared to the GTX 1050 available on the personal development machine. This hardware enabled faster simulation throughput and allowed more efficient training of reinforcement learning policies.

5.3 Discussion

The use of multiple computing platforms during this research provided flexibility in conducting experiments under different hardware constraints. During the initial stages of development, many experiments were executed on CPU-based systems due to limited availability of high-performance GPU resources. In this context, MuJoCo proved particularly valuable because of its efficient CPU-based physics simulation, allowing reinforcement learning experiments to be performed reliably on standard workstation hardware.

In addition to MuJoCo, the Brax physics engine was also used across different hardware configurations. Since Brax is implemented using the JAX numerical computing framework, it can execute on CPUs as well as accelerator hardware such as GPUs. This flexibility allowed the same simulation framework to be used during both early experimentation and later large-scale training runs.

When GPU resources were available, Brax enabled more efficient experimentation through parallel simulation of multiple environments. This capability increased the rate at which training data could be generated, thereby reducing the overall training time for reinforcement learning policies.

Overall, the combination of CPU-efficient simulation through MuJoCo and the hardware-flexible design of Brax allowed the research workflow to adapt to the available computational resources while maintaining a consistent experimental pipeline.

5.4 Summary

This chapter described the simulation frameworks and computational infrastructure used for training and evaluating reinforcement learning agents in this research. Two

physics engines, MuJoCo and Brax, were employed to simulate control tasks and support reinforcement learning experimentation.

MuJoCo provided an efficient and stable simulation environment optimized for CPU-based computation, which was particularly useful during early development stages and on systems with limited GPU capabilities. Brax complemented this workflow by offering a flexible simulation framework capable of running on both CPU and GPU hardware through the JAX ecosystem. This allowed the training pipeline to scale efficiently when accelerator resources were available.

The experiments were conducted across three different computing platforms: a local development machine, an NVIDIA-based compute server, and a dedicated workstation equipped with an RTX 3060 GPU. Using multiple systems enabled the experiments to continue throughout the research period despite varying availability of computational resources, while still supporting both CPU-based experimentation and GPU-accelerated training.

Chapter 6

Development of the Hovering Policy

As discussed in Chapter 2, the application of reinforcement learning (RL) in aerial robotics has predominantly focused on underactuated quadrotor platforms. While these approaches have demonstrated impressive capabilities such as aggressive maneuvering, agile flight, and trajectory tracking, the application of reinforcement learning to fully actuated aerial robots remains relatively unexplored.

Fully actuated aerial platforms offer several advantages compared to conventional quadrotors. In particular, the ability to independently generate forces and torques enables improved maneuverability, enhanced disturbance rejection, and the potential to perform aerial physical interaction tasks. However, this additional control authority also increases the dimensionality of the control problem and complicates the policy learning process.

The literature review highlighted that robust low-level control policies are essential for enabling higher-level aerial behaviors. In particular, stable hovering represents a fundamental prerequisite for aerial manipulation and physical interaction tasks. Without a reliable hovering controller, more complex behaviors such as contact interaction or cooperative manipulation cannot be safely executed.

Motivated by these observations, this chapter presents the development of a reinforcement learning policy capable of stabilizing a fully actuated aerial robot in hovering conditions. Due to the limited availability of prior work in this area, the controller was developed from scratch while drawing inspiration from existing reinforcement learning approaches for quadrotor control.

Rather than directly attempting to learn a complex hovering controller, the policy development followed an incremental training strategy. The learning task was progressively increased in complexity, beginning with simple velocity damping behavior and gradually extending toward full three-dimensional position stabilization. This staged learning approach enabled stable training and provided important insights into the interaction between reward design, system dynamics, and reinforcement learning optimization.

The overall reinforcement learning training framework used in this work is illustrated in Figure 6.1. The control policy receives the system observation vector and generates

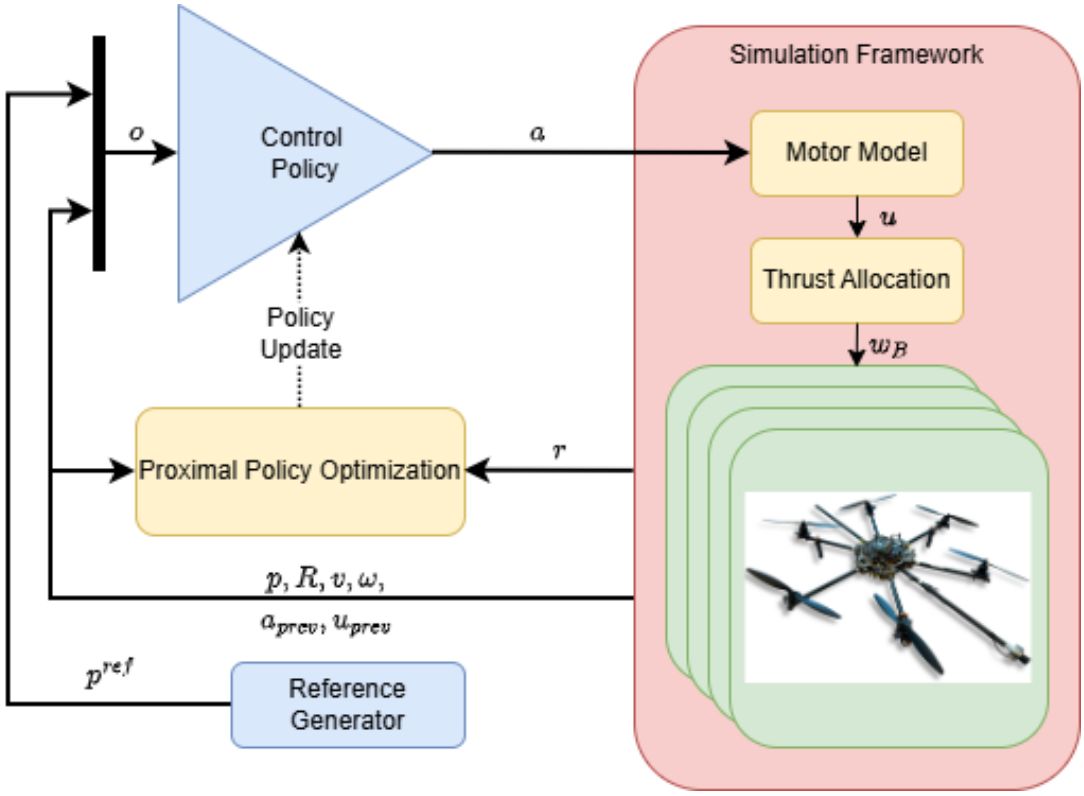

Figure 6.1: Overview of the reinforcement learning control framework used for training the hovering policy

control commands which are passed through a motor model and thrust allocation module before being applied to the simulated aerial robot. The resulting system state is used to compute a reward signal that is utilized by the Proximal Policy Optimization (PPO) algorithm to update the policy parameters.

6.1 Policy Development Strategy

Developing a reinforcement learning controller for a fully actuated aerial robot presents several challenges. Directly training a policy for full hovering using complex reward formulations proved to be unstable during the initial experiments. Early attempts using direct reward shaping for hovering did not converge to stable policies.

To address this issue, a progressive learning strategy was adopted. Instead of learning the hovering behavior directly, the reinforcement learning policy was first trained to imitate the behavior of simple classical controllers. This approach allowed the policy to gradually acquire the fundamental stabilization behaviors required for hovering.

The development process consisted of the following stages:

1. Velocity damping using PID-inspired behavior

2. Altitude tracking

3. Full position tracking

4. Final hovering policy refinement

Each stage introduced additional complexity to the control problem while building upon the behaviors learned in the previous stage.

6.2 Velocity Damping Policy

The first stage of policy development focused on learning velocity-damping behavior. In this stage, the objective of the policy was to reduce the linear velocity of the aerial robot along all three translational directions.

This behavior mimics the functionality of a classical PID velocity damping controller, which acts to stabilize the vehicle by reducing undesired motion. Training the policy to imitate this behavior allowed the reinforcement learning agent to learn the basic relationship between state observations and control actions. Basically testing the successfull brax environment.

6.2.1 Observation Space

The observation vector is defined as

$$o_t = \left[p^T, vec(R_B)^T, v^T, \omega^T, a_{prev}, u_{prev}\right]^T \tag{6.1}$$

where

- p is the robot position in the world frame
- R_B is the orientation of the robot body
- v is the linear velocity
- ω is the angular velocity
- a_{prev} represents the previous action
- u_{prev} represents previously realized thrusts

All matrix quantities are flattened using column-wise vectorization before being concatenated into the observation vector.

Including previous actions in the observation helps the policy maintain temporal smoothness and reduces high-frequency control oscillations.

6.2.2 Reward Function

To guide the learning process, a composite reward function is defined. The reward is designed to encourage accurate positioning, force tracking, smooth control actions, and stable robot orientation.

The overall reward at timestep t is defined as

$$r = r_{vel} + r_{act} + r_{flat} + r_{term} \tag{6.2}$$

Each term in the reward function is weighted by a scaling coefficient.

Velocity Reward

This error function encourages the drone to damp its velocity to zero

$$r_{vel} = \lambda_1 \left(\|v\|_2 \right) \tag{6.3}$$

Action Smoothness Reward

To reduce abrupt control signals, a smoothness penalty is applied to the change in actions.

$$r_{act} = \lambda_2 \|a - a_{prev}\|^2 \tag{6.4}$$

Orientation Alignment Reward

To maintain stable contact with the surface, the robot should remain approximately level during interaction. Therefore, an orientation alignment reward is introduced to penalize deviations from the desired orientation.

The orientation alignment reward is defined as

$$r_{flat} = \lambda_3 \left\| \frac{1}{2} \left(R_d^T R_B - R_B^T R_d \right)^V \right\|^2 \tag{6.5}$$

where R_B denotes the current body rotation matrix of the robot and R_d represents the desired orientation. The operator $(\cdot)^V$ converts the skew-symmetric matrix into its corresponding vector form.

In this work, the desired orientation is chosen as $R_d = I_3$ which corresponds to a flat orientation of the drone with respect to the world frame. This reward term penalizes orientation deviations and encourages the robot to maintain a stable posture during contact interaction.

Termination Penalty

A termination penalty is applied when an episode ends prematurely due to unsafe conditions.

$$r_{term} = \lambda_4 \cdot \mathbb{1}_{terminated} \tag{6.6}$$

Episode Termination Conditions

Training episodes terminate when unsafe or invalid robot states occur. The termination conditions include:

- Robot orientation exceeding 60° from upright
- Robot flies out of the $10m^3$ cube

These constraints ensure that the learning process remains within safe operational limits.

6.2.3 Training Results

The velocity damping policy was evaluated by analyzing the evolution of training metrics and the dynamic response of the aerial robot during simulation rollouts.

Figure 6.2 presents the training progress in terms of episode reward and average episode length. During the early stages of training, the policy exhibits large variance in the episode reward, with several unstable trajectories resulting in large negative returns. As training progresses, the reward gradually improves and stabilizes, indicating that the agent learns to reduce velocity errors and maintain stable flight behavior.

In addition to the reward trend, the average episode length also increases significantly during training. The episode length grows from a few hundred simulation steps at the beginning of training to nearly 5000 steps once the policy stabilizes. This indicates that the learned controller is capable of maintaining stable behavior for extended durations without triggering termination conditions.

To further evaluate the learned controller, a rollout simulation was performed using the trained policy. Figure 6.3 shows the translational velocity response of the aerial robot along the three body axes. The controller effectively dampens the initial velocity disturbances and drives the system toward a near-zero velocity state. After the initial transient phase, the velocities remain close to zero throughout the remainder of the simulation, demonstrating stable damping behavior.

The orientation evolution of the drone during the same rollout is shown in Figure 6.4. Small attitude adjustments occur during the initial transient period as the controller compensates for the velocity disturbances. After stabilization, the roll, pitch, and yaw

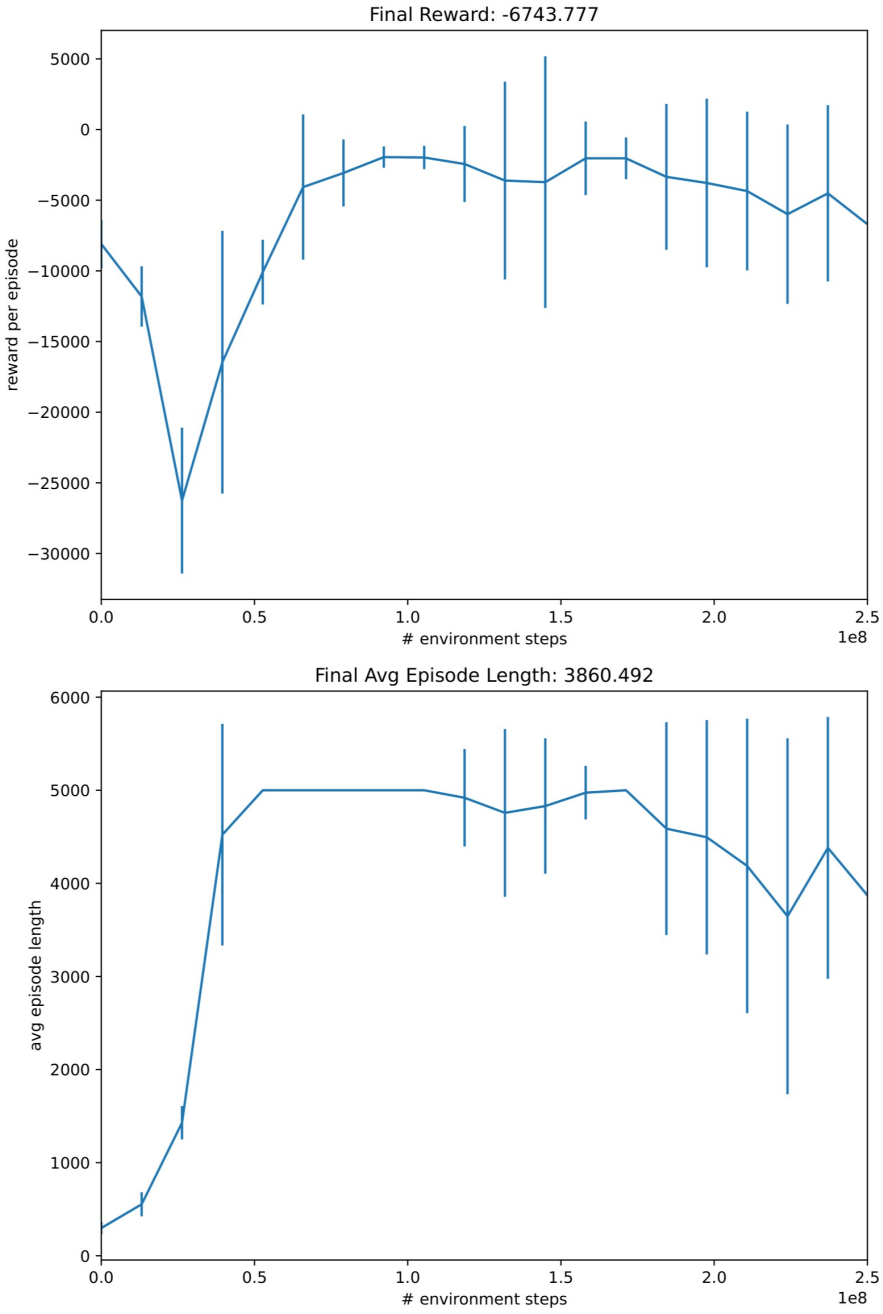

Figure 6.2: Training progress of the velocity damping policy showing the evolution of episode reward and average episode length over training steps.

angles remain close to constant values. In particular, the pitch angle remains within approximately 0.5° of its equilibrium value during steady-state operation, indicating that the velocity stabilization behavior does not introduce large orientation deviations.

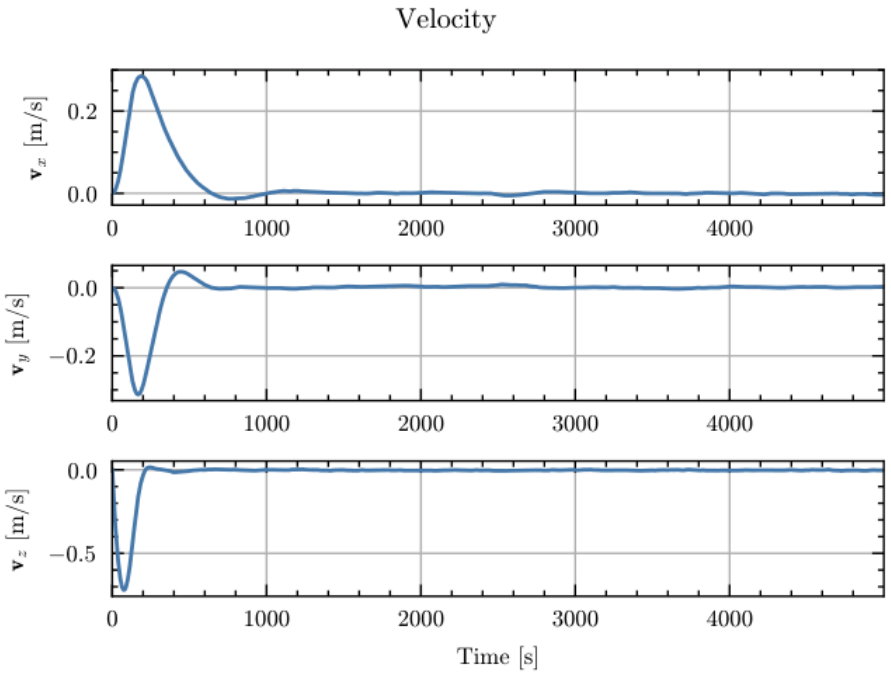

Figure 6.3: Velocity response of the drone during rollout using the velocity damping policy. The controller suppresses velocity disturbances along all three axes.

The control actions generated by the learned policy are illustrated in Figure 6.5. The action signals initially exhibit larger variations while the controller compensates for the initial disturbances. Once the system stabilizes, the control commands become smoother and remain within a relatively narrow range. This behavior indicates that the policy learns to produce consistent control inputs required to maintain the damped velocity state.

It should be noted that the final stages of training exhibit signs of potential overtraining, where continued training does not significantly improve the performance metrics. In such cases, selecting the best-performing policy checkpoint during training may provide a more representative controller than the final trained model.

Overall, the results demonstrate that the reinforcement learning policy successfully learns the fundamental behavior of velocity damping. The policy effectively suppresses translational motion and stabilizes the drone's velocity, providing a reliable foundation for the subsequent stages of altitude and position control learning.

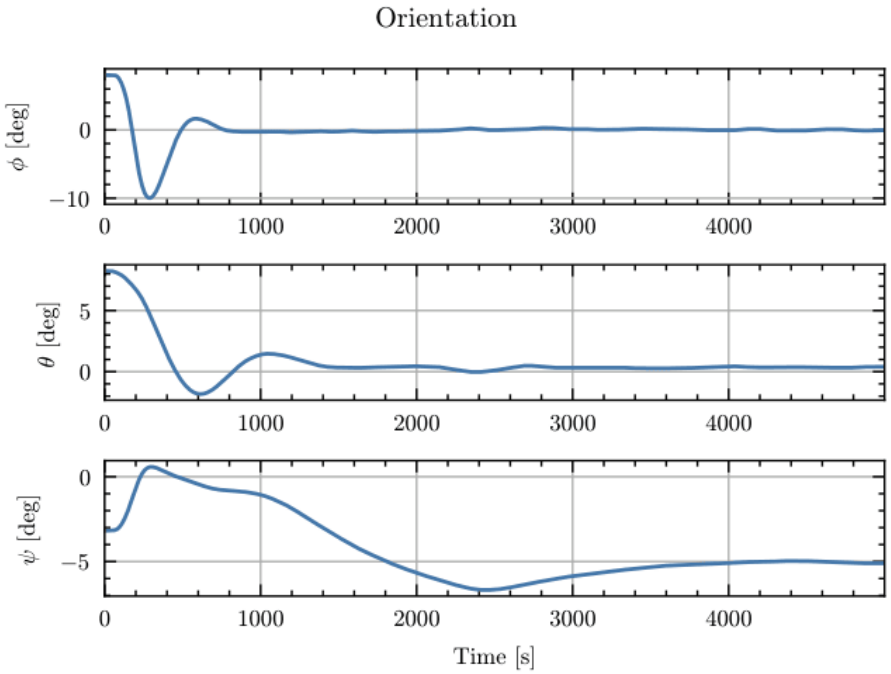

Figure 6.4: Orientation evolution of the drone during velocity damping. The controller maintains a nearly constant orientation after the initial transient phase.

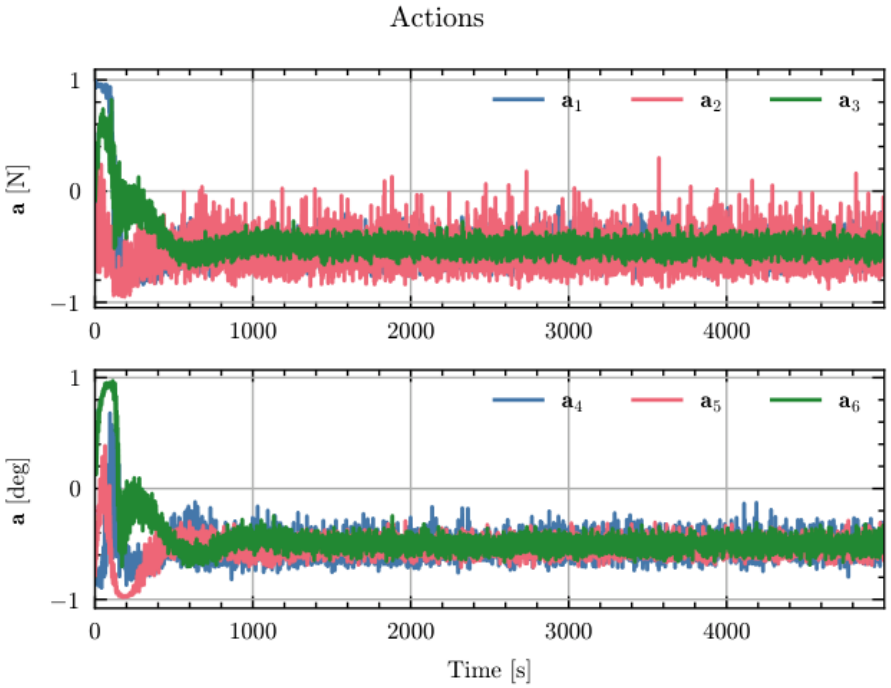

Figure 6.5: Control actions generated by the velocity damping policy during rollout. The commands stabilize after the initial transient period.

6.3 Altitude Tracking Policy

After achieving stable velocity damping, the next stage introduced altitude control. In this task, the policy was required to maintain a desired vertical position while continuing to damp velocity disturbances.

Altitude stabilization is a critical component of hovering control. By learning this task separately, the policy was able to develop an understanding of the thrust dynamics required to counteract gravity while maintaining vertical stability.

6.3.1 Observation Space

The observation vector is defined as

$$o_t = \left[p^T, vec(R_B)^T, v^T, \omega^T, p_B^{ref}, a_{prev}, u_{prev}\right]^T \tag{6.7}$$

where

- p is the robot position in the world frame
- R_B is the orientation of the robot body
- v is the linear velocity
- ω is the angular velocity
- p_B^{ref} is the desired position in body frame
- f^{ref} is the desired normal contact force
- a_{prev} represents the previous action
- u_{prev} represents previously realized thrusts

All matrix quantities are flattened using column-wise vectorization before being concatenated into the observation vector.

Including previous actions in the observation helps the policy maintain temporal smoothness and reduces high-frequency control oscillations.

6.3.2 Reward Function

To guide the learning process, a composite reward function is defined. The reward is designed to encourage accurate positioning, force tracking, smooth control actions, and stable robot orientation.

The overall reward at timestep t is defined as

$$r = r_{pos} + r_{vel} + r_{act} + r_{flat} + r_{term} \tag{6.8}$$

Each term in the reward function is weighted by a scaling coefficient.

Position Tracking Reward

The coarse position reward encourages the Drone to move toward the desired reference altitude i.e position in Z direction only.

$$r_{pos} = \lambda_1 \left(1 - \tanh\left(\frac{\|p_Z - p_Z^{ref}\|_2}{\sigma}\right)\right) \tag{6.9}$$

Velocity Reward

This error function encourages the drone to damp its velocity to zero

$$r_{vel} = \lambda_2 \left(\|v\|_2\right) \tag{6.10}$$

Action Smoothness Reward

To reduce abrupt control signals, a smoothness penalty is applied to the change in actions.

$$r_{act} = \lambda_3 \|a - a_{prev}\|^2 \tag{6.11}$$

Orientation Alignment Reward

To maintain stable contact with the surface, the robot should remain approximately level during interaction. Therefore, an orientation alignment reward is introduced to penalize deviations from the desired orientation.

The orientation alignment reward is defined as

$$r_{flat} = \lambda_4 \left\| \frac{1}{2} \left(R_d^T R_B - R_B^T R_d\right)^V \right\|^2 \tag{6.12}$$

where R_B denotes the current body rotation matrix of the robot and R_d represents the desired orientation. The operator $(\cdot)^V$ converts the skew-symmetric matrix into its corresponding vector form.

In this work, the desired orientation is chosen as $R_d = I_3$ which corresponds to a flat orientation of the drone with respect to the world frame. This reward term penalizes orientation deviations and encourages the robot to maintain a stable posture during contact interaction.

Termination Penalty

A termination penalty is applied when an episode ends prematurely due to unsafe conditions.

$$r_{term} = \lambda_5 \cdot \mathbb{1}_{terminated} \tag{6.13}$$

Episode Termination Conditions

Training episodes terminate when unsafe or invalid robot states occur. The termination conditions include:

- Robot orientation exceeding 60° from upright
- Robot flies out of the $10m^3$ cube

These constraints ensure that the learning process remains within safe operational limits.

6.3.3 Training Results

The altitude tracking policy was evaluated using both training metrics and rollout simulations to analyze the learned control behavior. The training performance is illustrated in Figure 6.6, which shows the evolution of episode reward and average episode length throughout the learning process.

During the initial phase of training, the episode reward remains close to zero while the agent explores the environment and begins to learn basic stabilization behaviors. As training progresses, the reward rapidly increases, indicating that the policy successfully learns to approach and maintain the desired altitude. The average episode length also increases steadily and eventually reaches the maximum episode length of 5000 steps. This indicates that the agent is able to maintain stable flight conditions for the full duration of an episode without triggering termination conditions.

To further evaluate the learned policy, a rollout simulation was performed using the trained controller. The resulting position trajectories are shown in Figure 6.7. The drone initially starts with a significant altitude error relative to the reference position. The controller gradually reduces this error and converges to the desired altitude. Once the reference altitude is reached, the position remains close to the desired value with minimal steady-state deviation.

The velocity response of the drone during the rollout is presented in Figure 6.8. The initial transient motion results in larger velocity magnitudes as the controller attempts to reach the reference altitude. After the altitude error is reduced, the velocities gradually

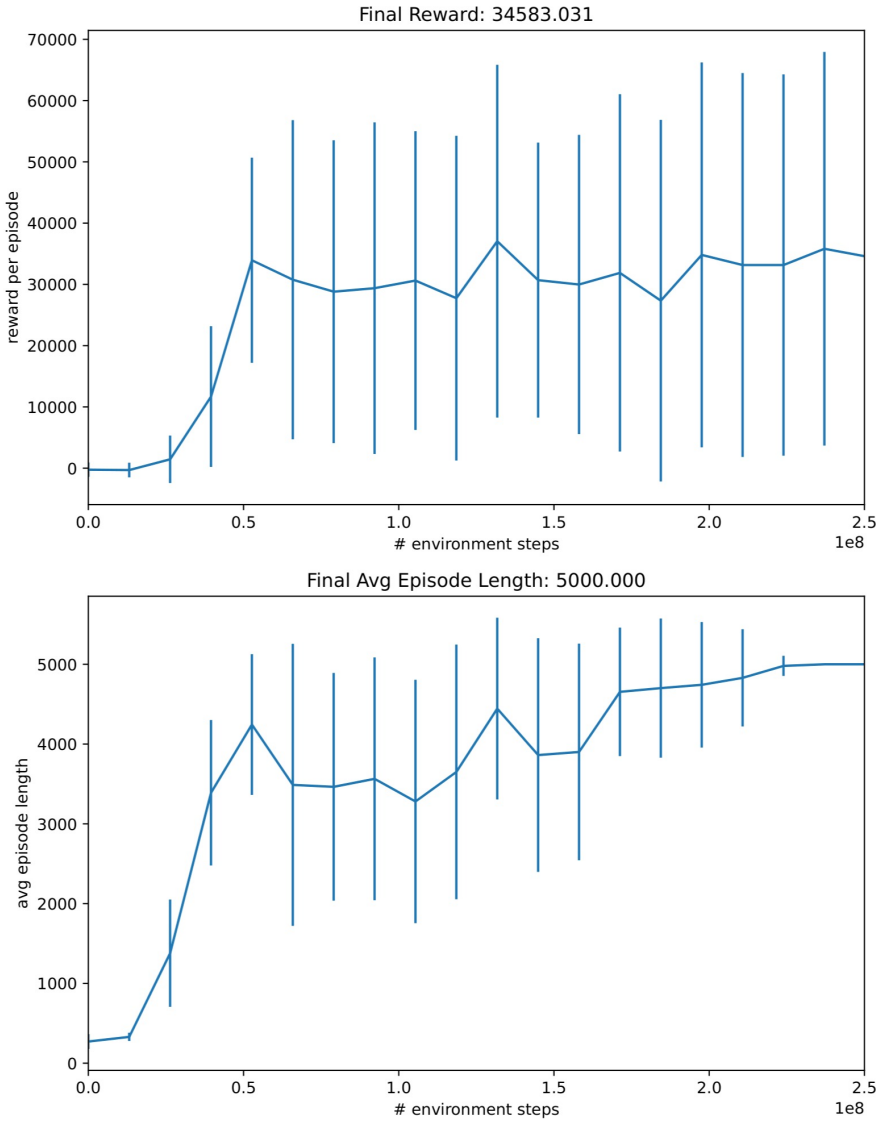

Figure 6.6: Training progress of the altitude tracking policy showing the evolution of episode reward and average episode length.

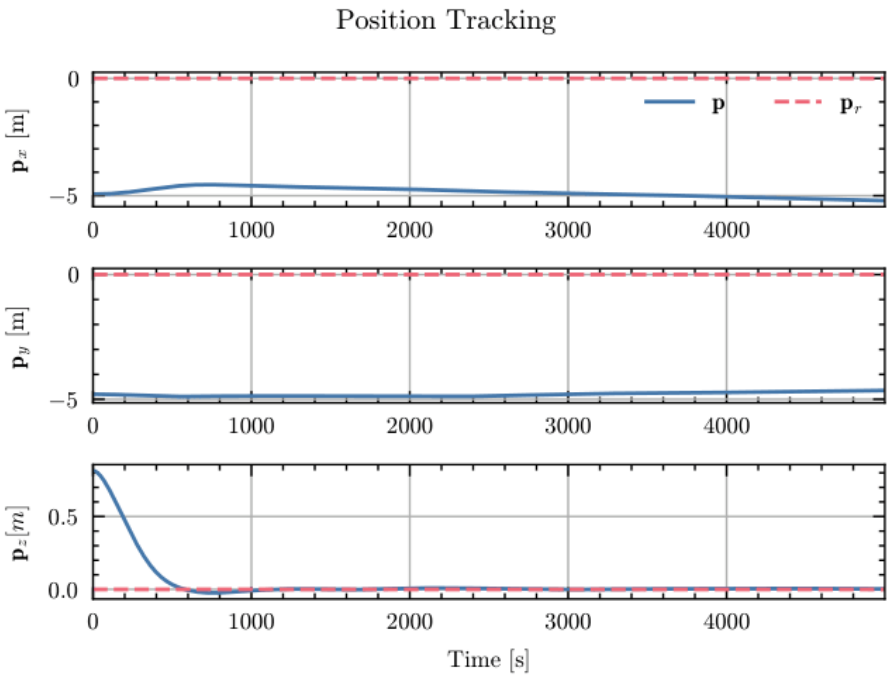

Figure 6.7: Position response of the drone during altitude tracking. The vertical position converges to the desired reference altitude.

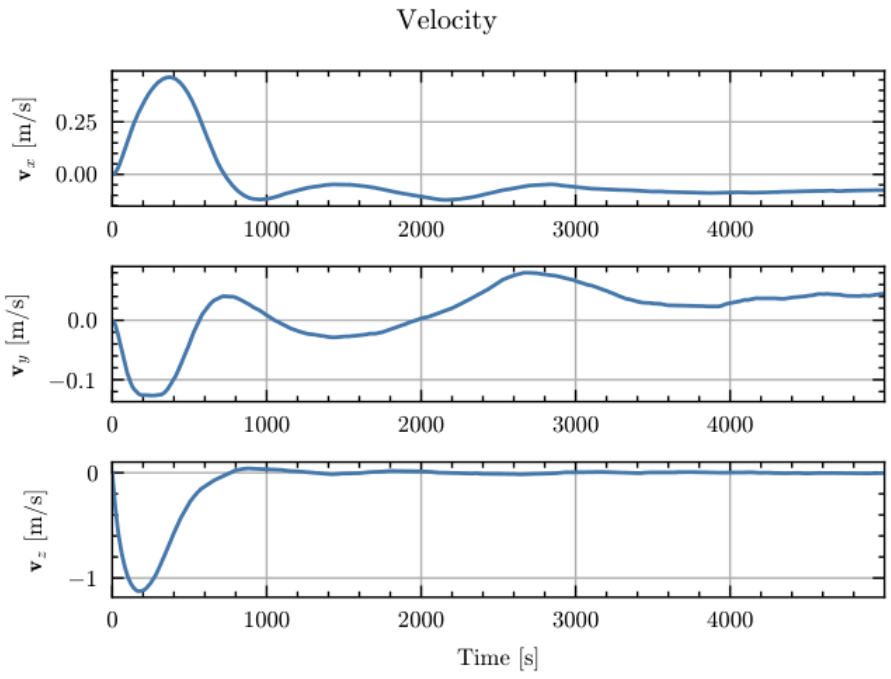

Figure 6.8: Velocity evolution during altitude tracking. The controller damps the translational velocity once the desired altitude is reached.

converge toward zero, indicating that the drone successfully stabilizes its motion around the target position.

Figure 6.9 shows the evolution of the drone's orientation during the altitude tracking experiment. The roll and pitch angles exhibit moderate transient responses during the initial stabilization phase as the controller generates corrective thrust to move toward the desired altitude. After the transient period, the orientation gradually stabilizes and remains within a bounded range, indicating that the altitude controller does not introduce excessive attitude deviations.

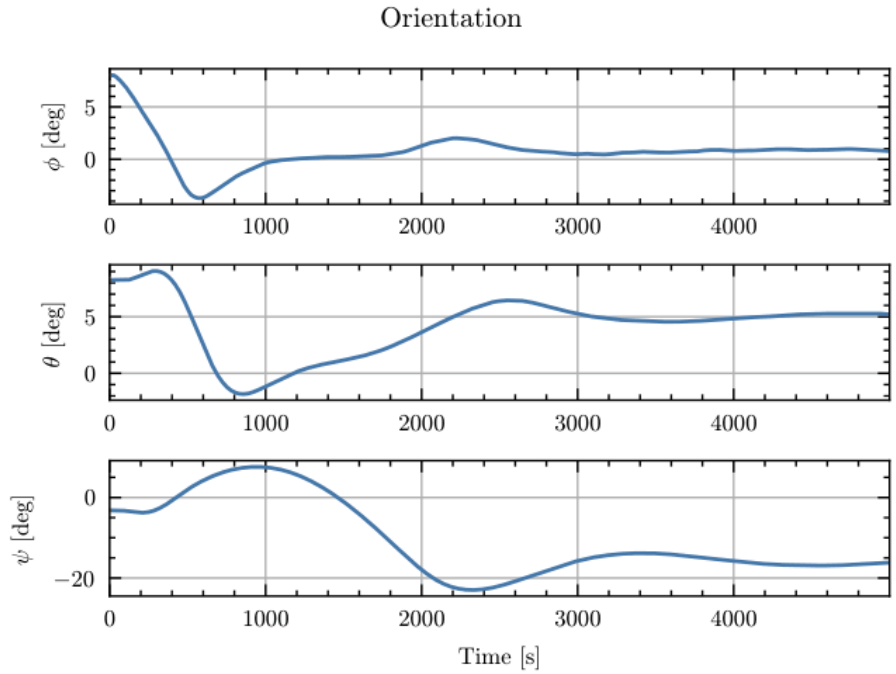

Figure 6.9: Orientation evolution of the drone during altitude tracking. The attitude stabilizes after the initial transient phase.

The control actions generated by the learned policy are illustrated in Figure 6.10. Larger variations in control inputs are observed during the initial transient phase as the controller compensates for the altitude error. Once the drone reaches the desired altitude, the control commands become smoother and remain within a relatively narrow operating range. This indicates that the policy learns to generate consistent control inputs required for maintaining the desired vertical position.

Overall, the results demonstrate that the reinforcement learning policy successfully learns to regulate the drone's altitude while maintaining stable flight dynamics. The agent is able to reach the desired vertical position and maintain it while damping residual velocities, providing a stable foundation for extending the controller toward full three-dimensional position tracking.

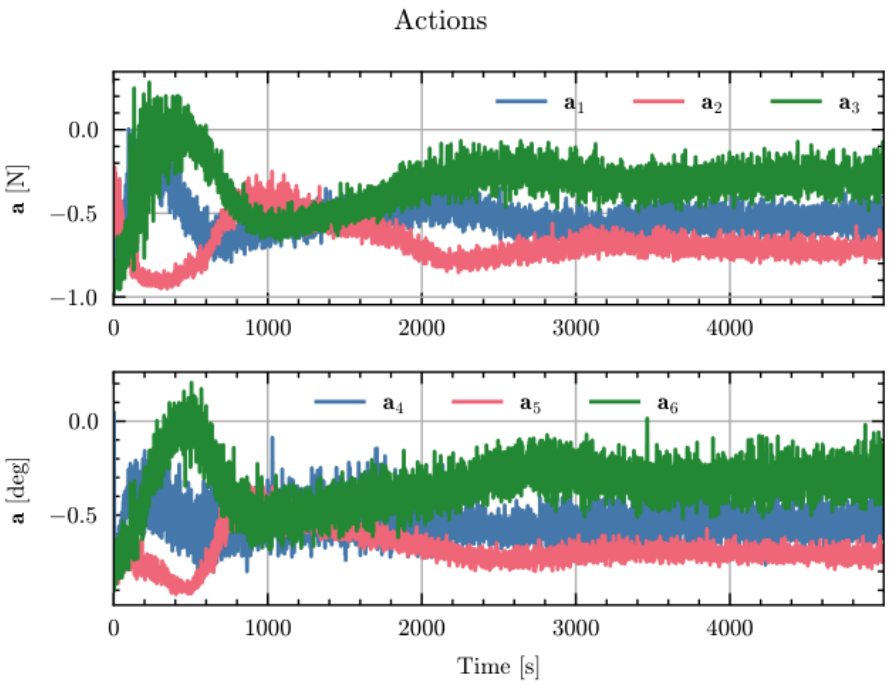

Figure 6.10: Control actions generated by the altitude tracking policy during rollout.

6.4 Position Tracking Policy

Following the successful development of the altitude tracking controller, the next objective was to extend the policy to full three–dimensional position tracking. In this stage, the reinforcement learning agent was trained to reach and maintain a desired spatial position in the environment.

Position tracking introduces additional complexity compared to altitude regulation, as the controller must simultaneously regulate motion along all three translational axes while maintaining a stable orientation. The policy must therefore learn to coordinate thrust generation and attitude adjustments to achieve the desired position while suppressing undesired velocities.

During training, the policy demonstrated the ability to move toward the desired reference position. However, it was observed that although the drone was capable of reaching the target location, it struggled to remain stationary at the desired position. After reaching the reference, the drone exhibited small oscillations and drift behavior, preventing it from maintaining a perfectly stable hover.

6.4.1 Observation Space

The observation vector provided to the policy is defined as

$$o_t = \left[p^T, \mathrm{vec}(R_B)^T, v^T, \omega^T, p_B^{ref}, a_{prev}, u_{prev}\right]^T \tag{6.14}$$

where

- p represents the robot position in the world frame
- R_B denotes the rotation matrix describing the body orientation
- v is the linear velocity of the robot
- ω represents the angular velocity
- p_B^{ref} is the desired reference position expressed in the body frame
- a_{prev} represents the previous control action
- u_{prev} represents the previously realized thrust values

All matrix quantities are flattened using column-wise vectorization before being concatenated into the observation vector.

Including the previous control action in the observation vector helps the policy learn smoother control behavior and reduces high-frequency oscillations in the generated commands.

6.4.2 Reward Function

To guide the learning process, a composite reward function was defined to encourage accurate position tracking, velocity damping, smooth control actions, and stable robot orientation.

The overall reward at timestep t is defined as

$$r = r_{pos} + r_{vel} + r_{act} + r_{flat} + r_{term} \tag{6.15}$$

Each component is scaled by a corresponding weighting coefficient.

Position Tracking Reward

The primary reward component encourages the robot to minimize the position error relative to the desired reference location.

$$r_{pos} = \lambda_1 \left(1 - \tanh\left(\frac{\|p - p^{ref}\|_2}{\sigma}\right)\right) \tag{6.16}$$

This reward increases as the robot approaches the desired position.

Velocity Damping Reward

A velocity penalty is included to encourage the drone to reduce translational velocity and remain stationary near the target location.

$$r_{vel} = \lambda_2 \|v\|_2 \tag{6.17}$$

Action Smoothness Reward

To reduce abrupt control commands, a smoothness penalty is applied to changes in the control action.

$$r_{act} = \lambda_3 \|a - a_{prev}\|^2 \tag{6.18}$$

Orientation Alignment Reward

To maintain a stable posture, the robot is encouraged to remain close to a flat orientation.

$$r_{flat} = \lambda_4 \left\| \frac{1}{2} \left(R_d^T R_B - R_B^T R_d \right)^V \right\|^2 \tag{6.19}$$

where R_B is the current body orientation and R_d represents the desired orientation. The operator $(\cdot)^V$ converts the skew-symmetric matrix into vector form.

In this work, the desired orientation is defined as

$$R_d = I_3 \tag{6.20}$$

which corresponds to a level orientation of the drone with respect to the world frame.

Termination Penalty

A termination penalty is applied when an episode ends due to unsafe conditions.

$$r_{term} = \lambda_5 \mathbb{1}_{terminated} \tag{6.21}$$

Episode Termination Conditions

Training episodes terminate when unsafe robot states occur. The termination conditions include

- robot orientation exceeding 60° from upright
- the robot leaving the predefined $10\,m^3$ operational volume

These constraints ensure that training remains within safe operational limits.

6.4.3 Training Results

The training progress of the position tracking policy is illustrated in Figure 6.11. During the initial phase of training, the episode reward remains relatively small while the agent explores the environment. As learning progresses, the reward increases significantly, indicating that the policy learns to approach the desired reference position. The average episode length also increases steadily and eventually approaches the maximum episode duration, suggesting that the drone is able to maintain stable flight for extended periods without triggering termination conditions.

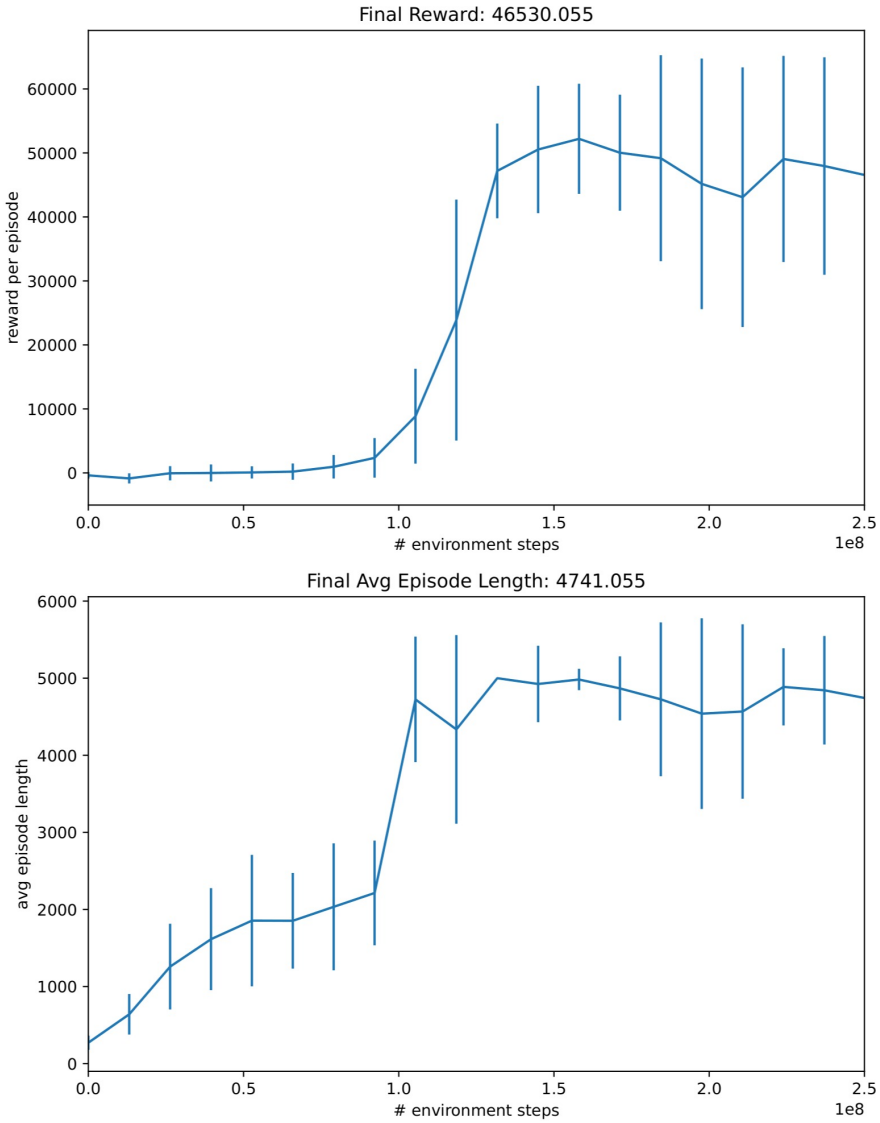

Figure 6.11: Training progress of the position tracking policy showing episode reward and average episode length.

To evaluate the behavior of the learned controller, a rollout simulation was performed using the trained policy. The resulting position trajectories are shown in Figure 6.12. The drone initially starts with a large position error and moves toward the desired reference

location. The controller successfully drives the drone toward the target position; however, after reaching the reference point, the system exhibits oscillatory behavior and small drift around the desired position.

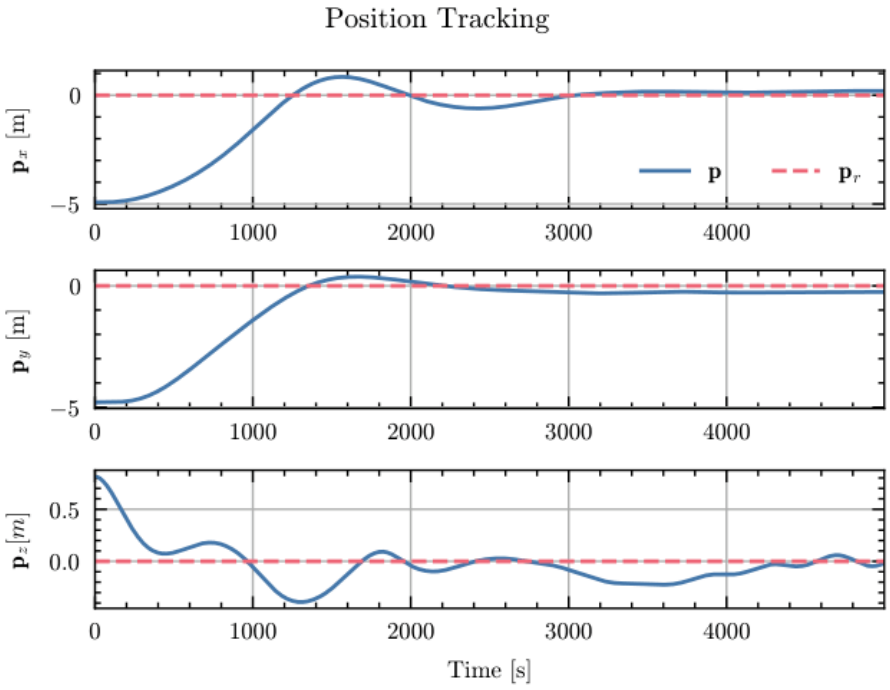

Figure 6.12: Position response of the drone during position tracking. The drone approaches the reference position but exhibits oscillatory behavior around the target.

The velocity evolution during the rollout is presented in Figure 6.13. Large velocity magnitudes occur during the initial transient phase as the drone moves toward the target position. Although the velocity decreases after the reference position is reached, small oscillations remain present in the steady-state regime.

Figure 6.14 shows the orientation evolution of the drone during the same rollout experiment. The roll and pitch angles exhibit moderate deviations during the transient motion as the drone accelerates toward the target location. After the transient phase, the orientation stabilizes but still exhibits slow variations corresponding to the oscillatory motion observed in the position trajectories.

The control actions generated by the learned policy are illustrated in Figure 6.15. The controller initially produces large actuation commands to accelerate toward the reference position. Once the target is reached, the actions become smaller but continue to fluctuate as the controller attempts to compensate for the residual oscillations around the reference location.

Overall, the results demonstrate that the reinforcement learning policy successfully learns to navigate toward the desired spatial location. However, maintaining a perfectly stable position remains challenging, as the system exhibits small oscillations around the reference point. This behavior highlights the sensitivity of reinforcement learning control

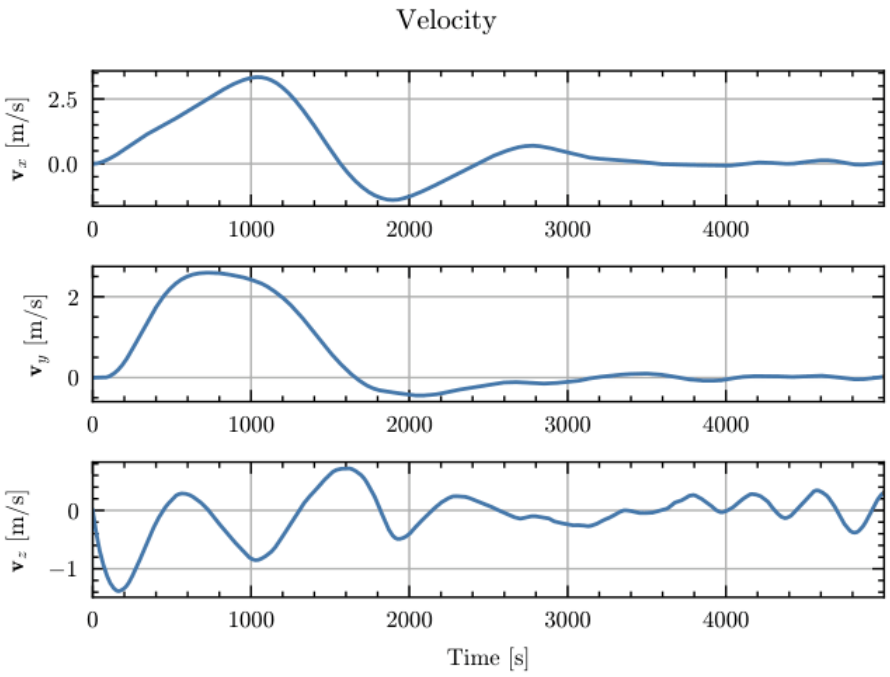

Figure 6.13: Velocity response of the drone during position tracking. Residual oscillations remain after reaching the reference position.

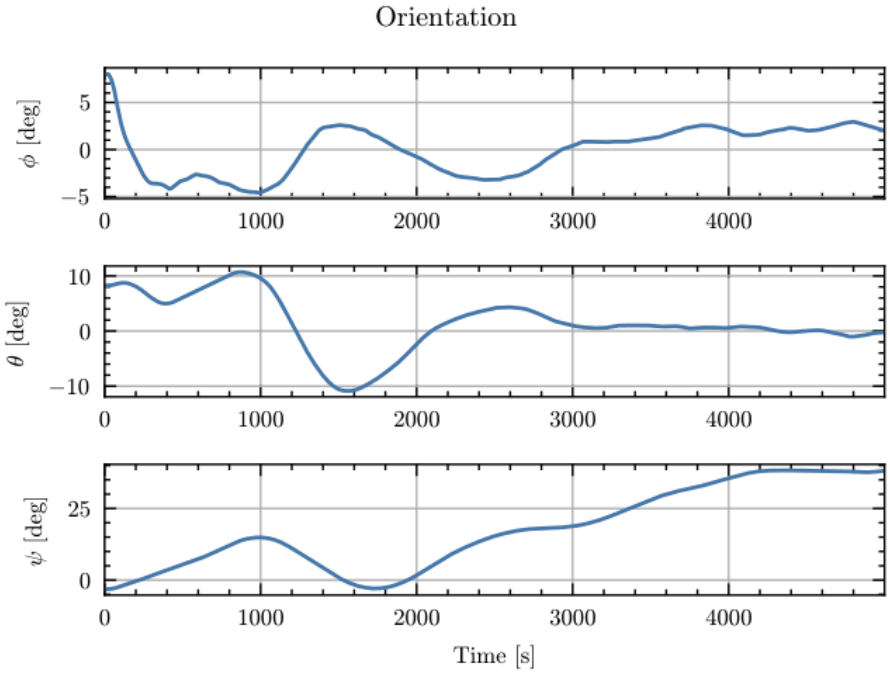

Figure 6.14: Orientation evolution during position tracking. The attitude stabilizes after the transient motion but exhibits slow variations due to position oscillations.

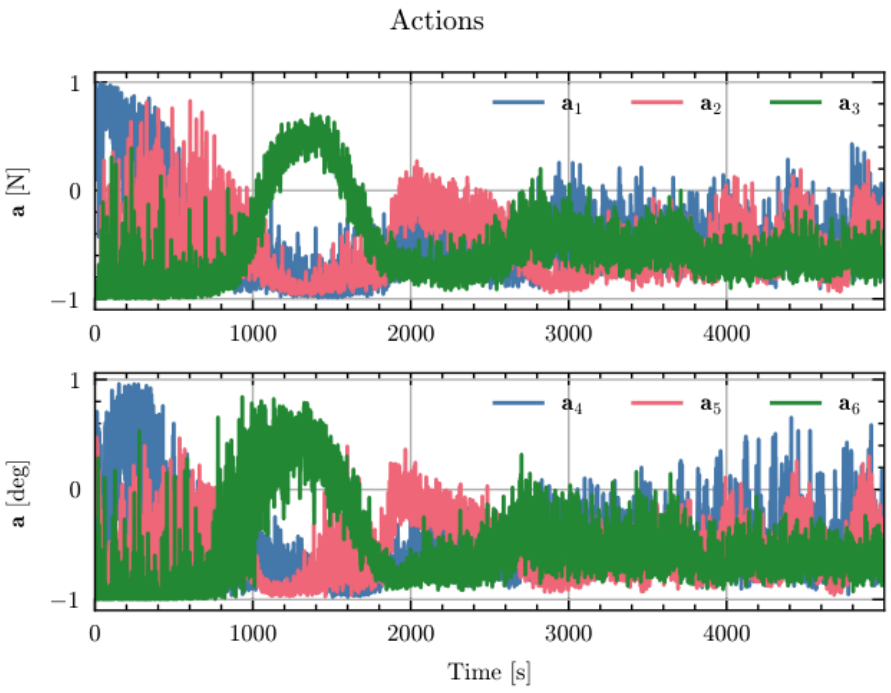

Figure 6.15: Control actions generated by the position tracking policy.

performance to reward design and system dynamics, motivating further refinement of the reward formulation and training strategy for achieving stable hovering behavior.

6.4.4 Observed Limitations

Although the position tracking policy was able to guide the drone toward the desired reference position, it was observed that the system struggled to maintain a stable hover once the target position was reached. Instead of settling at the desired location, the drone exhibited oscillatory motion and slow drift around the reference point.

A detailed analysis revealed that this instability was primarily related to the formulation of the velocity reward term. While the velocity penalty encouraged the policy to minimize translational velocity, it occasionally produced control actions that conflicted with the stabilization behavior required to maintain position. In certain situations, the policy generated corrective actions that reduced instantaneous velocity but unintentionally pushed the drone away from the equilibrium position.

As a result, the controller repeatedly applied compensating actions that prevented the system from converging to a steady hover. This behavior highlights an important challenge in reinforcement learning based control, namely the sensitivity of policy learning to reward design and control frequency.

These observations motivated further investigation into the training configuration and the numerical properties of the simulation environment.

6.5 Training Improvements

During subsequent experimentation, two key factors were identified that significantly improved training stability and final policy performance: the simulation timestep and the policy update frequency.

6.5.1 Simulation Timestep

The first factor influencing training stability was the simulation timestep. Initial experiments were performed using a timestep of

$$\Delta t = 0.002$$

Under this configuration, the training process frequently produced unstable policies and slow convergence. Small changes in control inputs produced large variations in the system response, which made policy optimization difficult.

Increasing the timestep slightly to

$$\Delta t = 0.0025$$

resulted in significantly improved training stability. This modification altered the numerical integration behavior of the simulator and produced smoother state transitions. As a consequence, the reinforcement learning algorithm was able to learn more consistent control strategies and converge more reliably.

6.5.2 Policy Update Frequency

The second major improvement involved modifying the frequency at which the policy generated new control actions. In the original training configuration, a new action was produced at every simulation step.

To improve stability, the policy update frequency was reduced by repeating each action for multiple simulation steps. In the final configuration, the control policy operated at a frequency of approximately 200 Hz, meaning that each action generated by the policy was applied for two consecutive simulation steps before a new action was computed.

This modification provided two important benefits:

- It reduced the variance of policy updates during training.
- It improved the consistency and smoothness of the applied control inputs.

Together, these changes significantly improved the learning dynamics and allowed the reinforcement learning agent to converge to a stable hovering controller.

6.6 Final Hovering Policy

After incorporating the improvements in simulation timestep and policy update frequency, the training process was repeated for the hovering task. Under these conditions, the reinforcement learning agent successfully learned a stable hovering policy capable of maintaining the drone at a desired spatial position.

The resulting controller demonstrated smooth control actions and stable flight behavior across extended simulation durations. The learned policy was able to regulate position, suppress velocity disturbances, and maintain a stable orientation simultaneously.

6.6.1 Observation Space

The observation vector provided to the policy is defined as

$$o_t = \left[p^T, \text{vec}(R_B)^T, v^T, \omega^T, p_B^{ref}, a_{prev}, u_{prev}\right]^T \tag{6.22}$$

where

- p represents the robot position in the world frame
- R_B denotes the body rotation matrix
- v is the linear velocity
- ω represents the angular velocity
- p_B^{ref} is the desired reference position in the body frame
- a_{prev} denotes the previous control action
- u_{prev} represents the previously realized thrust values

All matrix quantities are vectorized before concatenation to form the final observation vector. Including the previous control action helps the policy learn smoother control strategies and reduces high-frequency oscillations in the generated commands.

6.6.2 Reward Function

To guide the learning process, a composite reward function was designed to encourage accurate position tracking, smooth control behavior, and stable robot orientation.

The overall reward is defined as

$$r = r_{pos} + r_{pos\_fine} + r_{act} + r_{flat} + r_{term} \tag{6.23}$$

Each component of the reward function is weighted using a scaling coefficient.

Coarse Position Reward

The primary reward encourages the drone to approach the desired reference position.

$$r_{pos} = \lambda_1 \left(1 - \tanh\left(\frac{\|p - p^{ref}\|_2}{\sigma_1}\right)\right) \tag{6.24}$$

Fine Position Reward

A secondary shaping reward improves positioning accuracy when the drone is close to the target.

$$r_{pos\_fine} = \lambda_2 \left(1 - \tanh\left(\frac{\|p - p^{ref}\|_2}{\sigma_2}\right)\right) \tag{6.25}$$

This term provides a stronger gradient near the target location and improves precision during hovering.

Action Smoothness Reward

To prevent abrupt control commands, a smoothness penalty is applied to the change in actions:

$$r_{act} = \lambda_3 \|a - a_{prev}\|^2 \tag{6.26}$$

Orientation Alignment Reward

To maintain stable flight, the drone is encouraged to remain close to a level orientation:

$$r_{flat} = \lambda_4 \left\| \frac{1}{2} \left(R_d^T R_B - R_B^T R_d\right)^V \right\|^2 \tag{6.27}$$

where $R_d = I_3$ represents the desired flat orientation.

Termination Penalty

Episodes terminate when unsafe conditions occur. A penalty is applied as

$$r_{term} = \lambda_5 \cdot \mathbb{1}_{terminated} \tag{6.28}$$

6.6.3 Training Results

The training progress of the hovering policy across multiple random seeds is illustrated in Figure 6.16. The reward curves show consistent improvement during training for several seeds, indicating that the learning process converges toward effective hovering behavior.

Table 6.1: Reward weighting coefficients used for training the hovering policy.

Reward Term	Description	Weight Value
λ_1, σ_1	Coarse position reward	15.0, 2.0
λ_2, σ_2	Fine position reward	15.0, 0.2
λ_3	Action smoothness penalty	-0.01
λ_4	Orientation alignment penalty	-10
λ_5	Termination penalty	-500.0

In addition, the average episode length increases steadily and approaches the maximum episode duration, suggesting that the drone remains stable throughout most training episodes.

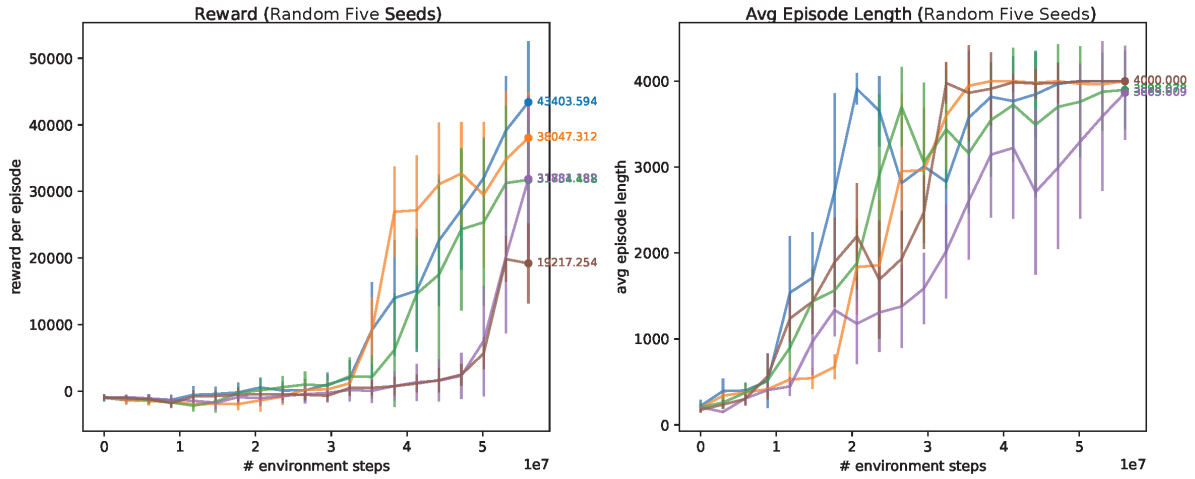

Figure 6.16: Training performance of the hovering policy across multiple random seeds.

To further evaluate the learned policy, rollout simulations were conducted using the trained controller. The resulting position trajectories are shown in Figure 6.17. The drone quickly converges toward the desired reference position and maintains a stable hover with minimal steady-state error.

The velocity evolution during the rollout is illustrated in Figure 6.18. After the initial transient motion, the translational velocities converge close to zero, indicating successful suppression of disturbances and stable hovering behavior.

The orientation evolution is presented in Figure 6.19. The roll and pitch angles exhibit small transient deviations during stabilization but quickly converge to near-constant values. This demonstrates that the learned controller maintains a stable attitude while performing position regulation.

Finally, the control actions generated by the policy are shown in Figure 6.20. After the initial stabilization phase, the control signals remain smooth and bounded, indicating that the learned policy generates consistent and physically plausible control inputs.

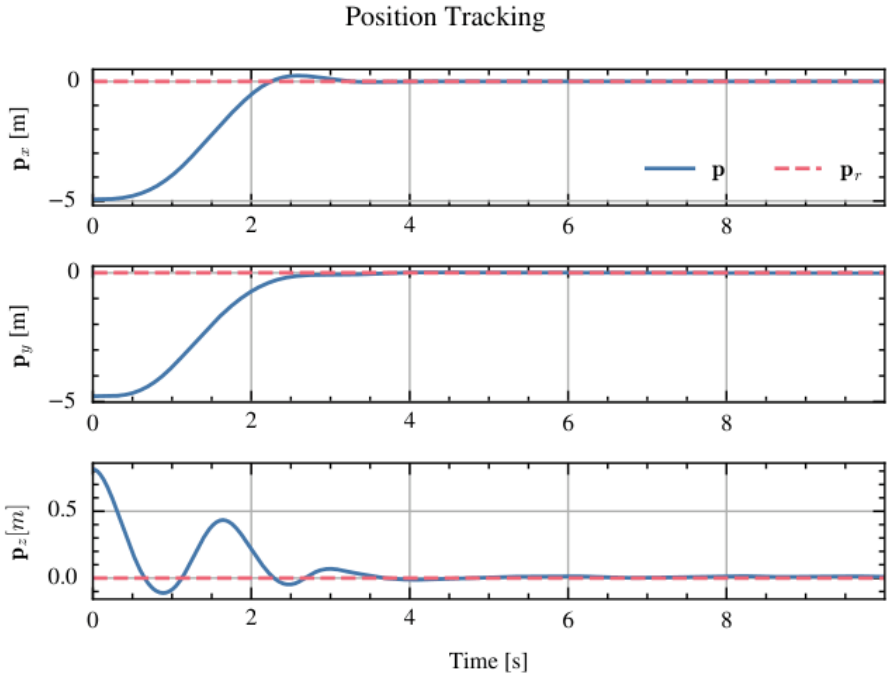

Figure 6.17: Position response of the drone during hovering. The controller maintains the reference position with small steady-state error.

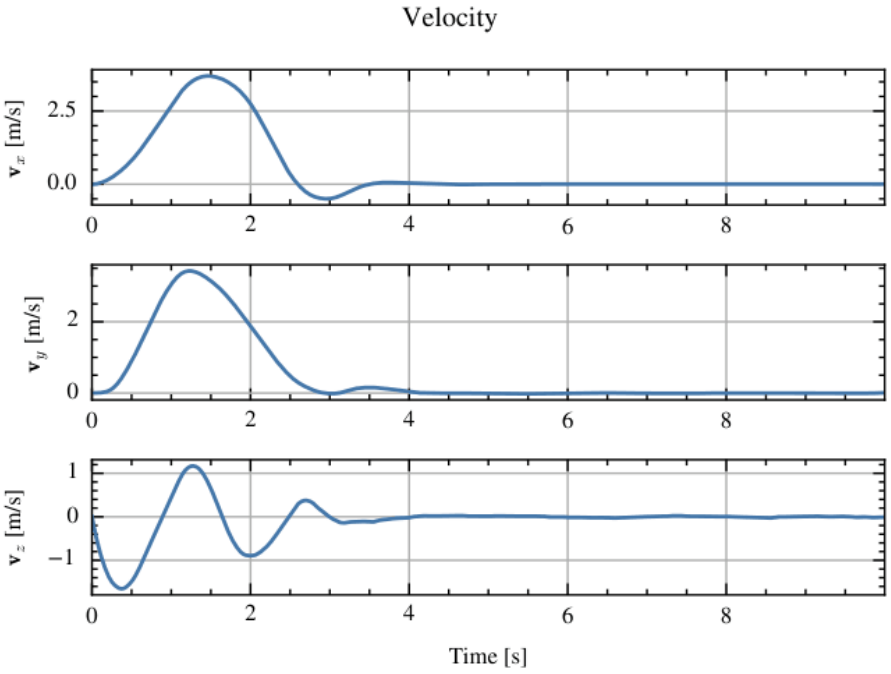

Figure 6.18: Velocity response of the drone during hovering.

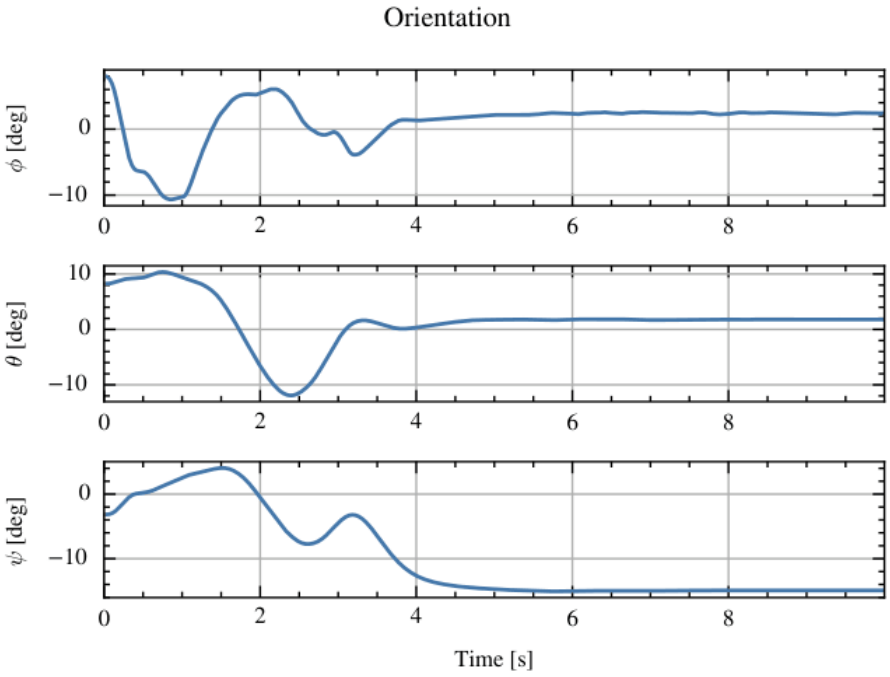

Figure 6.19: Orientation evolution during hovering.

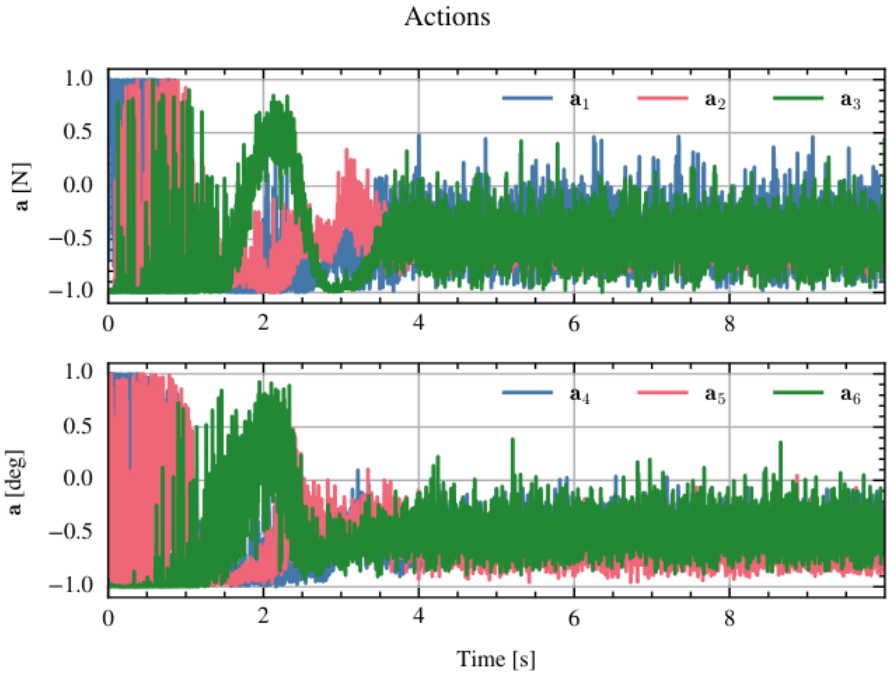

Figure 6.20: Control actions generated by the hovering policy.

Overall, the results demonstrate that the reinforcement learning controller is capable of maintaining stable hovering behavior while simultaneously regulating position, velocity, and orientation.

6.7 Discussion

The experiments presented in this chapter provide several insights into reinforcement learning based control of fully actuated aerial robots.

First, attempting to directly learn complex control tasks can lead to unstable training behavior. A progressive learning strategy, in which the controller is trained on increasingly complex tasks, can significantly improve learning stability.

Second, reward design plays a critical role in reinforcement learning performance. Even small inconsistencies between reward terms can introduce conflicting control objectives that prevent policy convergence.

Finally, system-level parameters such as simulation timestep and policy update frequency strongly influence training dynamics. Careful tuning of these parameters is essential for obtaining stable and reliable policies.

6.8 Conclusion

This chapter presented the development of a reinforcement learning based hovering controller for a fully actuated aerial robot. Due to the limited availability of prior work in this area, the controller was developed using a progressive training strategy.

The policy was initially trained to perform velocity damping, followed by altitude tracking and full three-dimensional position control. During this process, several important insights were obtained regarding reward design, simulation timestep selection, and policy update frequency.

By incorporating these improvements, the reinforcement learning agent successfully learned a stable hovering policy capable of maintaining the aerial robot at a desired spatial position. This hovering controller serves as a fundamental building block for enabling more advanced aerial interaction tasks, which will be explored in the following chapters.

Chapter 7

Contact Modeling and Design Simulations

7.1 Contact Modeling for Aerial Manipulation

In aerial manipulation tasks, a drone equipped with an end-effector must interact with external surfaces while maintaining flight stability. Unlike ground robots, aerial platforms are highly sensitive to external disturbances due to their underactuated dynamics and limited thrust margins. Therefore, accurate modeling of contact interactions between the end-effector and the environment is essential for ensuring stable and reliable interaction.

In this work, we model the contact interaction between the end-effector attached to a fully actuated drone and a vertical wall using the MuJoCo physics engine. The objective of the contact model is to enable the drone to exert a controlled normal force against the wall while maintaining stable contact during static interaction and sliding motions.

Two tasks are considered for validating the contact model:

1. Maintaining contact with a constant normal force.

2. Sliding along the wall while maintaining a constant normal force.

The goal of the modeling process is to determine suitable contact solver parameters that allow stable interaction with minimal penetration, limited slipping, and acceptable force oscillations.

7.1.1 Soft Contact Modeling in MuJoCo

MuJoCo uses a soft-contact formulation in which small penetrations between bodies are allowed and contact forces are generated to restore the separation between objects [11]. Instead of enforcing rigid non-penetration constraints, MuJoCo models contact as a compliant interaction between bodies.

When the end-effector comes into contact with the wall, the contact force can be approximated using a spring-damper model:

$$F_n = kd + b\dot{d} \tag{7.1}$$

where

- d is the penetration depth
- $\dot{d}$ is the penetration velocity
- k is the effective contact stiffness
- b is the damping coefficient

This formulation ensures that the contact dynamics remain smooth and numerically stable, which is important for simulation-based analysis and controller design.

The compliant formulation also ensures that the simulation remains stable even during rapid contact transitions such as when the drone approaches the wall or begins sliding along the surface.

7.1.2 Contact Pair Definition

In the simulation model, the contact interaction occurs between the geom representing the end-effector and the geom representing the wall surface.

The contact pair is explicitly defined to ensure that contact interactions are only evaluated between the end-effector and the wall. This reduces unnecessary collision checks and improves simulation efficiency.

```
<contact>
    <pair geom1="end_effector" geom2="wall"/>
</contact>
```

This configuration ensures that the drone body does not unintentionally collide with the wall and that only the end-effector participates in contact interactions.

7.1.3 Contact Solver Parameters

The behavior of the contact interaction is controlled by several solver parameters in MuJoCo. The parameters most relevant to this work are `solimp`, `solref`, and `impratio`.

solref

In this work, the negative format of the `solref` parameter is used. This format directly specifies the stiffness and damping of the constraint:

$$\texttt{solref} = [-k, -b] \tag{7.2}$$

where k is the stiffness parameter and b is the damping parameter.

For the final model configuration, the selected value is

$$\texttt{solref} = [-10000, -200] \tag{7.3}$$

This setting produces a sufficiently stiff contact that limits penetration while providing adequate damping to prevent excessive oscillations during contact.

solimp

The `solimp` parameter controls how the solver transitions between free motion and constrained contact. It defines the impedance characteristics of the contact constraint.

The final selected value is

$$\texttt{solimp} = [0.99\ 0.99\ 0.0001\ 0.5\ 2] \tag{7.4}$$

These values create a smooth constraint activation region while maintaining strong contact impedance when the end-effector presses against the wall.

impratio

The `impratio` parameter determines the relative weighting between frictional constraints and penetration constraints.

A larger value prioritizes the enforcement of friction constraints and helps reduce slipping during contact interactions.

The selected value is

$$\texttt{impratio} = 100 \tag{7.5}$$

This value was found to significantly reduce slipping during sliding tasks while maintaining stable contact.

7.1.4 Experimental Evaluation

To evaluate the effectiveness of the contact model, two experimental tasks were performed in simulation:

1. Maintaining constant contact with a specified normal force.

2. Sliding along the wall while maintaining a constant normal force.

The maximum normal force considered in the experiments was 5 N. This value corresponds to the maximum force that the drone's end-effector can exert on the wall without compromising the stability of the aerial platform.

The contact model was validated for forces of 3 N, 4 N, and 5 N.

Figure 7.1 illustrates the contact behavior during the experiment, including penetration depth, end-effector position, and contact forces over time.

The results show that contact is maintained throughout the interaction period while the drone applies a consistent normal force against the wall.

7.1.5 Results

From the experimental analysis, the following observations were made:

- The slipping error remained within the range of 0.15 mm to 0.3 mm over a period of 10 seconds.
- The penetration depth remained below 0.1 mm during most of the simulation.
- Even during dynamic contact transitions, the penetration never exceeded 0.25 mm.
- The contact interaction exhibits small oscillations due to the compliant contact model.
- The oscillation frequency is significantly higher than the controller cutoff frequency of 10 Hz, and therefore does not affect system stability.

These results indicate that the contact model is sufficiently stiff to prevent excessive penetration while still allowing smooth force regulation.

7.1.6 Observed Limitations

Although the contact model performs well for both tasks, several limitations were observed:

- During sliding in the Y direction, a slight loss of normal force occurs.
- The friction force measured by the force sensor in the Y direction shows noticeable oscillations.
- Some experimental trials exhibited larger transient oscillations during the initial contact phase.

However, despite these issues, the average force values remain consistent with the desired contact force, indicating that the overall model behavior remains valid.

7.1.7 Final Model Parameters

Based on extensive testing and parameter tuning, the following contact parameters were selected for the final simulation model:

$$\texttt{solimp} = [0.99\ 0.99\ 0.0001\ 0.5\ 2] \tag{7.6}$$

$$\texttt{solref} = [-10000\ -200] \tag{7.7}$$

$$\texttt{impratio} = 100 \tag{7.8}$$

These parameters provide stable contact behavior, minimal slipping, and acceptable penetration levels for aerial manipulation tasks involving wall interaction.

7.1.8 Conclusion

This chapter presented the development and validation of a contact model for a drone equipped with an end-effector interacting with a wall surface. The model was designed to support two key interaction tasks: maintaining a constant normal force and sliding along the wall while preserving contact.

Experimental evaluation demonstrated that the selected solver parameters produce stable contact interactions with minimal penetration and low slipping errors. Although small oscillations are observed in the force measurements, their frequency remains well above the control bandwidth and therefore does not affect the system's stability.

Based on these results, the selected contact parameters are considered suitable for aerial manipulation tasks involving wall interaction.

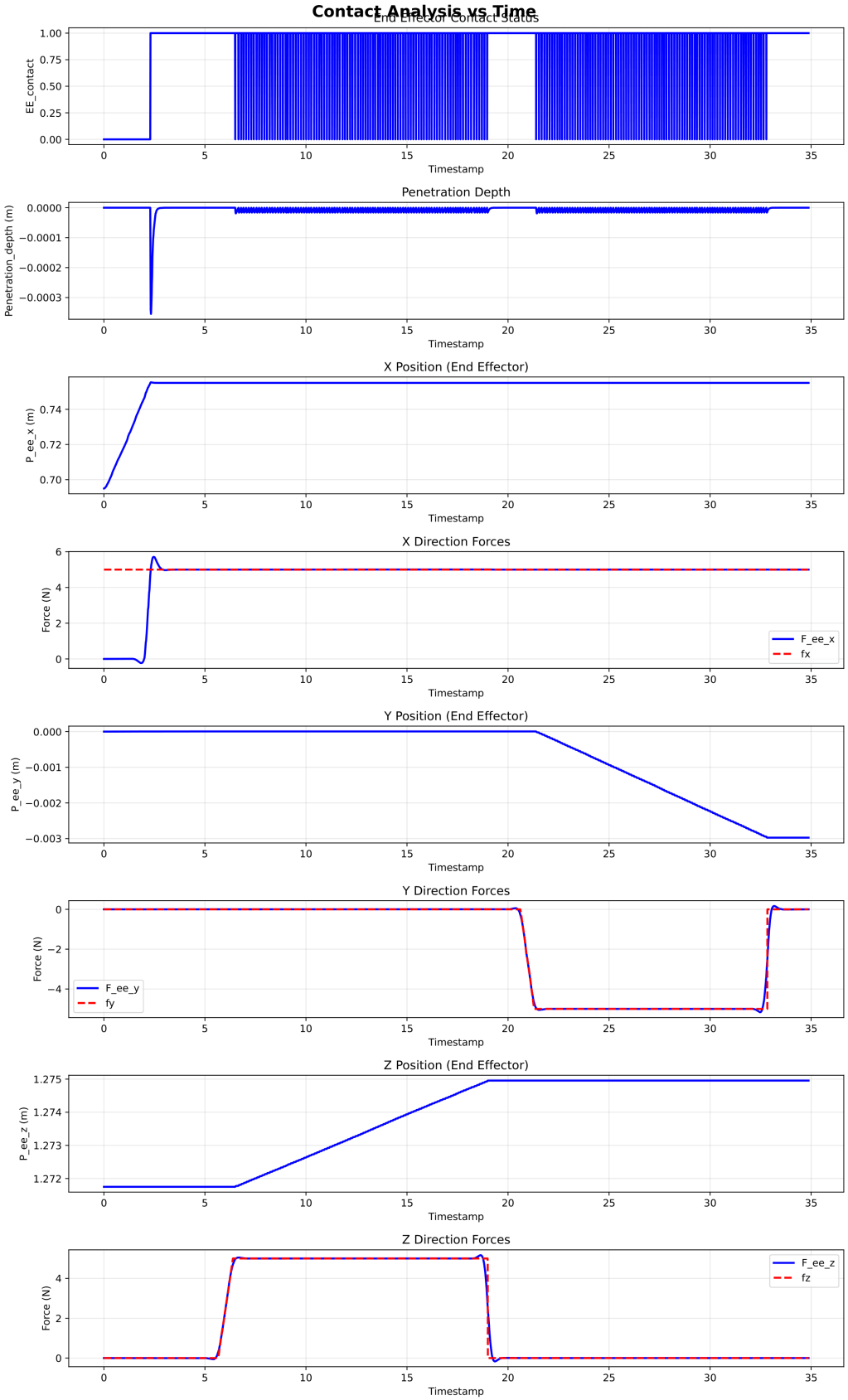

Figure 7.1: Contact analysis showing penetration depth, end-effector position, and contact forces during interaction with the wall.

Chapter 8

Conclusions

8.1 Summary of the Research

This thesis investigated the application of reinforcement learning for controlling a fully actuated aerial robot and explored its potential for enabling aerial physical interaction tasks. The motivation for this work arises from the growing need for aerial robots capable of performing not only perception tasks but also direct interaction with their environment.

While aerial robotics has seen significant advances in recent years, most research has focused on conventional quadrotor platforms performing free-flight tasks such as trajectory tracking, navigation, and aggressive maneuvering. In contrast, aerial physical interaction introduces additional challenges due to the coupling between flight dynamics and contact forces. These challenges are further amplified when considering fully actuated aerial platforms with higher control dimensionality.

To address these challenges, this thesis investigated reinforcement learning as a control paradigm capable of learning complex behaviors directly from interaction with a simulated environment. The research focused on developing the fundamental components required for enabling aerial interaction tasks.

8.2 Main Contributions

The primary contributions of this work can be summarized as follows.

First, a reinforcement learning framework for controlling a fully actuated aerial robot was developed. The control problem was formulated as a Markov Decision Process, enabling the design of observation spaces, action spaces, and reward functions tailored to aerial stabilization and interaction tasks.

Second, the dynamic model of a fully actuated hexarotor platform equipped with tilted rotors and an interaction end-effector was introduced. This model served as the foundation for simulation-based training and analysis of reinforcement learning policies.

Third, a progressive policy development strategy was proposed for learning stable hovering control. Rather than directly training a complex controller, the learning process was structured into incremental stages beginning with velocity damping, followed by altitude control, position tracking, and final hovering stabilization. This staged approach improved training stability and allowed the policy to gradually acquire the necessary control behaviors.

Fourth, simulation environments based on modern physics engines were utilized to enable large-scale reinforcement learning experiments. The use of MuJoCo and Brax allowed efficient training and evaluation of control policies while accurately modeling system dynamics and actuator constraints.

Finally, contact modeling experiments were performed to analyze the interaction behavior between the aerial robot and external surfaces. These simulations provided insights into how contact forces affect the system dynamics and how such interactions can be incorporated into reinforcement learning environments.

8.3 Implications for Aerial Physical Interaction

Although the primary focus of this thesis was the development of a stable hovering policy, the results provide important insights for future aerial physical interaction research.

Aerial interaction tasks require extremely stable flight performance, since the aerial robot must maintain precise position and orientation while experiencing external forces generated during contact. The reinforcement learning controller developed in this work demonstrated the ability to stabilize a fully actuated aerial platform and maintain controlled hovering behavior in simulation.

In addition, the dynamic modeling framework, reinforcement learning architecture, and simulation environments established in this thesis provide the essential infrastructure for studying contact-based aerial tasks.

Together, these components represent the fundamental building blocks required for aerial physical interaction. The stabilization policies developed in this work form the basis upon which more advanced interaction behaviors—such as force tracking, surface sliding, or manipulation—can be built.

Therefore, while full aerial manipulation was not the primary objective of this thesis, the research successfully establishes the necessary ingredients for enabling such capabilities in future work.

8.4 Limitations

Despite the promising results obtained in this research, several limitations should be acknowledged.

First, the reinforcement learning policies were trained and evaluated exclusively in simulation environments. Although modern physics simulators provide accurate dynamic modeling, transferring learned policies to real-world aerial platforms remains a challenging problem due to the well-known simulation-to-reality gap.

Second, the experiments focused primarily on hovering stabilization rather than complete aerial interaction tasks. While this focus was intentional—given that stable hovering is a prerequisite for interaction—it limits the direct demonstration of contact-based manipulation behaviors.

Third, reinforcement learning training remains computationally intensive and sensitive to reward design. Further research is required to develop more efficient learning strategies and improve policy robustness under varying environmental conditions.

8.5 Future Work

The results presented in this thesis open several promising directions for future research.

One important direction is the extension of the learned hovering controller to perform full aerial physical interaction tasks. This includes learning policies capable of regulating both motion and interaction forces during sustained contact with external surfaces.

Another avenue for future work is the transfer of learned reinforcement learning policies from simulation to real-world aerial platforms. Techniques such as domain randomization, sim-to-real transfer methods, and real-world fine-tuning could be explored to improve the practical applicability of the learned controllers.

Future research may also investigate more advanced reinforcement learning algorithms and training strategies to improve sample efficiency and policy robustness. Incorporating additional sensory modalities such as force–torque sensing, vision-based perception, or tactile feedback could further enhance the capability of aerial manipulation systems.

Finally, extending the framework to more complex aerial manipulation scenarios—such as cooperative manipulation, tool usage, or interaction with deformable environments—represents an exciting long-term research direction.

8.6 Final Remarks

This thesis presented an investigation into reinforcement learning control of a fully actuated aerial robot with the long-term goal of enabling aerial physical interaction tasks. Through the development of dynamic models, reinforcement learning frameworks, hovering control policies, and contact simulation environments, this work establishes the foundational components necessary for studying interaction-capable aerial robots.

The results demonstrate that reinforcement learning is a promising approach for controlling complex aerial systems and provide a solid basis for future research on aerial

manipulation and physical interaction. In this sense, the work presented in this thesis represents an important step *towards* reinforcement learning control of aerial robots capable of interacting with the physical world.

Bibliography

[1] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, 2017.

[2] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[3] R. Ferede, C. De Wagter, D. Izzo, and G. de Croon, "End-to-end reinforcement learning for time-optimal quadcopter flight," *arXiv preprint*, 2023.

[4] J. Eschmann, D. Albani, and G. Loianno, "Learning to fly in seconds," *IEEE Robotics and Automation Letters*, 2024.

[5] D. Zhang, A. Loquercio, J. Tang, T.-H. Wang, J. Malik, and M. W. Mueller, "A learning-based quadcopter controller with extreme adaptation," *arXiv preprint*, 2024.

[6] R. Ferede, T. Blaha, E. Lucassen, C. De Wagter, and G. de Croon, "One net to rule them all: Domain randomization in quadcopter racing across different platforms," *arXiv preprint*, 2025.

[7] E. Cuniato, O. Andersson, H. Oleynikova, R. Siegwart, and M. Pantic, "Learning to fly omnidirectional micro aerial vehicles with an end-to-end control network," in *Springer Proceedings in Advanced Robotics*, 2024.

[8] W. Zhang, L. Ott, M. Tognon, and R. Siegwart, "Learning variable impedance control for aerial sliding on uneven heterogeneous surfaces by proprioceptive and tactile sensing," *IEEE Robotics and Automation Letters*, 2022.

[9] Y. Feng, C. Shi, J. Du, Y. Yu, F. Sun, and Y. Song, "Variable admittance interaction control of uavs via deep reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, (London, UK), pp. 1291–1297, IEEE, May 2023.

[10] E. Cuniato, I. Geles, W. Zhang, O. Andersson, M. Tognon, and R. Siegwart, "Learning to open doors with an aerial manipulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[11] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[12] C. D. Freeman *et al.*, "Brax: A differentiable physics engine for large scale rigid body simulation," *arXiv preprint arXiv:2106.13281*, 2021.