POLITECNICO DI TORINO
Repository ISTITUZIONALE

Reinforcement Learning-Based PD Controller Gains Prediction for Quadrotor UAVs

Original
Reinforcement Learning-Based PD Controller Gains Prediction for Quadrotor UAVs / Sönmez, S., Montecchio, L., Martini, S., Rutherford, M.J., Rizzo, A., Stefanovic, M., Valavanis, K.P.. - In: DRONES. - ISSN 2504-446X. - 9:8(2025). [10.3390/drones9080581]

Availability:
This version is available at: 11583/3004843 since: 2025-11-05T16:07:20Z

Publisher:
MDPI

Published
DOI:10.3390/drones9080581

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

18 November 2026

drones

MDPI

Article

Reinforcement Learning-Based PD Controller Gains Prediction for Quadrotor UAVs

Serhat Sönmez [1,*,†], Luca Montecchio [2,†], Simone Martini [3,†], Matthew J. Rutherford [4], Alessandro Rizzo [2], Margareta Stefanovic [5] and Kimon P. Valavanis [5]

[1] Department of Electrical & Electronics Engineering, Istanbul Medeniyet University, 34700 Istanbul, Turkey
[2] Department of Electronics & Telecommunications, Politecnico di Torino, 10129 Torino, Italy; luca.montecchio@studenti.polito.it (L.M.); alessandro.rizzo@polito.it (A.R.)
[3] Aerospace Engineering & Engineering Mechanics Department, University of Cincinnati, Cincinnati, OH 45221, USA; marti6so@ucmail.uc.edu
[4] Department of Computer Sciences, University of Denver, Denver, CO 80208, USA; matthew.rutherford@du.edu
[5] Department of Electrical & Computer Engineering, University of Denver, Denver, CO 80210, USA; margareta.stefanovic@du.edu (M.S.); kimon.valavanis@du.edu (K.P.V.)
* Correspondence: serhat.sonmez@medeniyet.edu.tr
† Co-first authors, these authors contributed equally to this work.

Abstract

This paper presents a reinforcement learning (RL)-based methodology for the online fine-tuning of PD controller gains, with the goal of bridging the gap between simulation-trained controllers and real-world quadrotor applications. As a first step toward real-world implementation, the proposed approach applies a Deep Deterministic Policy Gradient (DDPG) algorithm—an off-policy actor–critic method—to adjust the gains of a quadrotor attitude PD controller during flight. The RL agent was initially trained offline in a simulated environment, using MATLAB/Simulink 2024a and the UAV Toolbox Support Package for PX4 Autopilots v1.14.0. The trained controller was then validated through both simulation and experimental flight tests. Comparative performance analyses were conducted between the hand-tuned and RL-tuned controllers. Our results demonstrate that the RL-based tuning method successfully adapts the controller gains in real time, leading to improved attitude tracking and reduced steady-state error. This study constitutes the first stage of a broader research effort investigating RL-based PID, LQR, MRAC, and Koopman-integrated RL-based PID controllers for real-time quadrotor control.

Keywords: reinforcement learning; multirotor UAVs; PD controller

check for updates

Academic Editor: Xingling Shao

Received: 8 June 2025
Revised: 6 August 2025
Accepted: 11 August 2025
Published: 16 August 2025

Citation: Sönmez, S.; Montecchio, L.; Martini, S.; Rutherford, M.J.; Rizzo, A.; Stefanovic, M.; Valavanis, K.P. Reinforcement Learning-Based PD Controller Gains Prediction for Quadrotor UAVs. *Drones* **2025**, *9*, 581. https://doi.org/10.3390/drones9080581

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Unmanned aerial vehicles (UAVs) have experienced tremendous growth in recent decades. They have been used in various civilian and public domain applications, such as power line inspection [1], mining area monitoring [2], wildlife conservation and monitoring [3], border protection [4], infrastructure and building inspection [5], and precision agriculture [6], among others. Multirotor UAVs and, in particular, quadrotors have become the most widely used aerial platforms, due to their vertical take-off and landing (VTOL) capabilities, efficient hovering, and overall flight effectiveness.

Although several conventional control techniques have been developed, implemented, and tested effectively for quadrotor navigation and control (via simulation studies, simulated experiments, and in real time), learning-based methodologies and algorithms have re-

cently gained significant momentum, because it has been shown that using them improves platform modeling, thus subsequently enhancing navigation and control effectiveness. A learning-based methodology offers alternatives to parameter tuning and estimation and to learning and understanding the working environment. To this end, several representative surveys on developing and adopting machine learning (ML), deep learning (DL), or reinforcement learning (RL) algorithms for UAV modeling and control have been published. Carrio et al. [7] focus on DL methods and their applications to UAVs. The studies covered in this survey show that DL is effectively used to solve control and navigation problems for UAVs. Also, various controller methods are used to collect the data for the training of the DL agents. Polydoros and Nalpantidis [8] cover model-based reinforcement learning applications in the robotics field, but they also provide a section for RL applications on UAVs. Their survey underscores how model-based approaches can improve control performance in UAV systems by enabling faster adaptation to dynamic environments with fewer real-world interactions. Choi and Cha [9] cover ML techniques on UAVs for autonomous flight. By examining both control mechanisms and perception components, such as object recognition, the authors highlight the substantial progress made in enabling UAVs to execute designated tasks more efficiently and resiliently in dynamic settings. Azar et al. [10] focus on deep reinforcement learning approaches for control and navigation tasks on drones. Brunke et al. [11] study learning-based control in robotic fields. They cover UAVs in their research, and they provide studies comparing learning-based controller and conventional controller performances. The authors' most recent survey [12] focuses on multirotor navigation and control based on online learning. On a more specific note, Yoo et al. [13] use hybrid RL controllers for a quadrotor. They implement PD-RL and LQR-RL low and high gains for trajectory tracking, maintaining constant gains and deploying the policy on a micro quadrotor platform. Mosweu et al. [14] employ a DDPG-based approach to adjust PD controller gains within a cascaded control structure for a simulated multirotor UAV. The approach in [14] is not deeply explained but appears similar to the author's prior work [15], which was published shortly beforehand, and which demonstrated the feasibility of RL-based PID parameter tuning and estimation for UAVs. Contrary to the existing literature, this work expands on [14,15] by proposing a training framework that allows for training of the RL agent offline and deploying it on a physical quadrotor UAV in outdoor flights for online controller gain fine-tuning.

The focus of this research was on the real-world implementation and evaluation of a reinforcement learning (RL)-based method for online tuning of PD controller gains, as originally proposed in [15]. This study serves as the first step in a broader research effort that aims to transition various RL-trained controllers—including the PID, LQR, MRAC, and Koopman-integrated PID approaches—from simulation to hardware deployment. Unlike previous simulation-based studies, this work validated the RL-tuned PD controller experimentally on a physical quadrotor UAV. The test platform used an "X" configuration quadrotor, differing from the less conventional "+" configuration adopted in [15], and tracking accuracy was assessed using a circular trajectory rather than the previously tested helix trajectory, due to the safety constraint of the available experimental facility. Accordingly, the RL agent was retrained to account for the updated quadrotor configuration and the new tracking task. While the implementation, at this stage of the research, employed a PD structure to avoid increased controller complexity due to integral windup, the underlying framework is applicable to both PD and PID controllers, and further investigation on full PID implementation with anti-windup strategies, as well as other adaptive control methods, will be the subject of future work.

This research makes four main contributions: (i) It demonstrates the real-world implementation of an RL-based method for fine-tuning the gains of a PD controller on a

quadrotor UAV, where the agent is trained in simulation and deployed on hardware without requiring a high-fidelity model that includes drag, gyroscopic effects, or sensor noise. This simplifies the training process while still enabling online adaptation. (ii) The study validates the RL-tuned controller across simulation and real-world outdoor flight tests, highlighting its capacity to adjust control gains in response to external disturbances and model mismatches during flight. (iii) It provides an effective training and development framework for RL-based UAV control strategies, which can be deployed on real hardware flight controller boards (e.g., the Pixhawk flight control unit) and does not require the use of companion computers. Moreover, in doing so, it features a realistic assessment of the challenges faced during sim-to-real transfer, such as the effects of unmodeled dynamics, limited sensor accuracy, and quantized actuator inputs, offering practical insights for future deployments. (iv) The results show that the RL-based control framework is adaptable and effective under different quadrotor configurations and task conditions (e.g., "X" configuration, circular trajectory), thus laying the groundwork for generalizable, platform-independent learning-based control strategies.

Section 2 provides notation and background information related to the mathematical model of the quadrotor, the PD controller with feedback linearization that is implemented, and the RL-based technique. Section 3 introduces the proposed RL-based fine-tuning strategy. Section 4 presents the RL agent environment setup and its implementation in real hardware. The training phase, numerical simulations, and experimental results are provided in Section 5. Section 6 presents our results analysis, and Section 7 concludes the paper.

2. Notation and Background Information

2.1. Quadrotor Mathematical Model

Figure 1 describes the quadrotor structure and reference frame adopted in this work, which is aligned to the PX4 Autopilot convention.

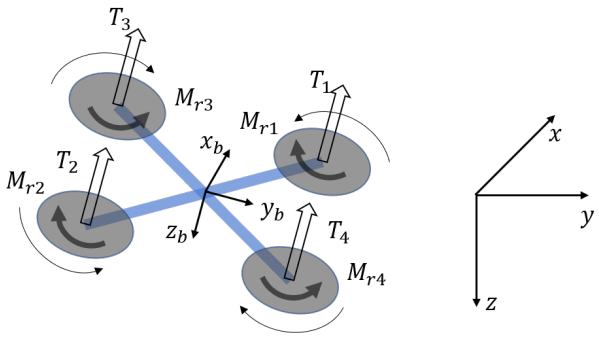


Figure 1. Quadrotor structure and reference frame.

Given two vectors, $a = [a_1, a_2, a_3]^\top$ and $b = [b_1, b_2, b_3]^\top$, the matrix $S(a)$ denotes the skew-symmetric matrix

$$S(a) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{1}$$

for which the following relation holds: $S(a)b = a \times b$.

The mathematical model of a quadrotor is derived by considering the "X" configuration, as opposed to the "+" configuration adopted in [15]. The Newton–Euler (N–E) equations of motion are given as follows:

$$\ddot{p} = g\hat{z} - \frac{1}{m} T \mathrm{R} \hat{z} \tag{2a}$$

$$\dot{R} = RS(\omega^B) \tag{2b}$$

$$I_f \dot{\omega}^B = -\omega^B \times (I_f \omega^B) + M_B \tag{2c}$$

Let $p = (x, y, z)^T$ express the position of the body-fixed frame B with respect to the inertial frame E. Gravitational acceleration and mass are defined by g and m, respectively. The unit vector along the z-axis is represented by $\hat{z} = (0, 0, 1)^T$. The rotation matrix $R \in SO(3)$ maps vectors from the body-fixed frame to the inertial reference frame, and it is parameterized using Euler angles $\eta = [\varphi, \theta, \psi]^\top$ as in [16],

$$R = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{3}$$

where $c_\alpha = \cos\alpha$, $s_\alpha = \sin\alpha$, and $\alpha \in \{\phi, \theta, \psi\}$; $\omega^B = [p, q, r]^\top$ represents angular velocities in the body-fixed frame, the relationship of which to the Euler rates is expressed by

$$\begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & \frac{s(\varphi)s(\theta)}{c(\theta)} & \frac{c(\varphi)s(\theta)}{c(\theta)} \\ 0 & c(\varphi) & -s(\varphi) \\ 0 & \frac{s(\varphi)}{c(\theta)} & \frac{c(\varphi)}{c(\theta)} \end{bmatrix}}_{W^{-1}} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4}$$

where W is defined as in [16]. I is the symmetric and positive-definite inertia matrix, which is computed with respect to the airframe's center of mass and expressed in the body-fixed frame. Finally, $M_B = [M_p, M_q, M_r]^T$ is the external torque expressed in the body-fixed frame.

Following [17], the quadrotor's model forcing terms are the results of a linear combination of each propeller thrust T_i,

$$T = (T_1 + T_2 + T_3 + T_4) \tag{5a}$$

$$M_p = \frac{\sqrt{2}}{2} l (T_2 + T_3 - T_1 - T_4) \tag{5b}$$

$$M_q = \frac{\sqrt{2}}{2} l (T_1 + T_3 - T_2 - T_4) \tag{5c}$$

$$M_r = \frac{c_D}{c_L} (T_1 + T_2 - T_3 - T_4) \tag{5d}$$

where l, c_D, and c_L denote the quadrotor arm length, the linear friction coefficient, and the angular friction coefficient, respectively. The thrust T_i is obtained through the spinning of the i-th propeller and controlled by a PWM signal to be sent to the electronic speed controller (ESC) following the control interface presented in [17]. To this end, it is desirable to express the forcing signals in terms of adimensional virtual control inputs τ_T, τ_R, τ_P, and τ_Y, which are dependent on the maximum input thrust T_{max} according to the following relations:

$$T = 4T_{max}(\tau_T - 1) \tag{6a}$$

$$M_p \frac{2}{l\sqrt{2}} = -4T_{max}\tau_R \tag{6b}$$

$$M_q \frac{2}{l\sqrt{2}} = -4T_{max}\tau_P \tag{6c}$$

$$M_r \frac{c_L}{c_D} = 4T_{max}\tau_Y \tag{6d}$$

2.2. Quadrotor Position and Attitude Controller

Quadrotors are treated as underactuated nonlinear systems, since position and attitude control (six generalized coordinates) are achieved through the four propellers' input thrust. A hierarchical control architecture has been followed for trajectory tracking tasks [18] that is composed of an outer position-control loop and an inner attitude-control loop, respectively.

2.2.1. Outer-Loop Control

The outer-loop control law is formulated independently for the z axis and x, y axis, respectively. The control law is achieved by inverting the position dynamics of (2a), and it has been slightly modified with respect to the one proposed in [17,19], since the drag effects have not been considered. To this end, the altitude control law is designed as

$$\tau_T = 1 - \frac{m}{4T_{max}(\cos\varphi)(\cos\theta)}(-g + v_z) \tag{7}$$

with v_z being the altitude virtual control law to be designed. Additionally, the desired Euler angles to achieve x, y-axis position control are computed as

$$\begin{bmatrix} \varphi_r \\ \theta_r \end{bmatrix} = F_B^{*-1} V_{xy} \tag{8}$$

with

$$F_B^* = -\frac{4T_{\max}(\tau_T - 1)}{m} \begin{bmatrix} \sin\psi & \cos\psi \\ -\cos\psi & \sin\psi \end{bmatrix} \tag{9}$$

and $V_{xy} = \begin{bmatrix} v_x, v_y \end{bmatrix}^\top$ being the x, y-position virtual control law to be designed.

The outer-loop outputs are τ_T, φ_r, θ_r. The τ_T goes to the PWM conversion block, while φ_r and θ_r along with the desired yaw angle ψ_r will be used as references for the inner-loop.

The proposed control laws are formulated as follows:

$$e_{\dot{x}} = k_{P1,x}(x_r - x) - \dot{x} \tag{10a}$$
$$e_{\dot{y}} = k_{P1,y}(y_r - y) - \dot{y} \tag{10b}$$
$$e_{\dot{z}} = k_{P1,z}(z_r - z) - \dot{z} \tag{10c}$$
$$v_x = k_{P2,x} e_{\dot{x}} + k_{D,x} \dot{e}_{\dot{x}} \tag{10d}$$
$$v_y = k_{P2,y} e_{\dot{y}} + k_{D,y} \dot{e}_{\dot{y}} \tag{10e}$$
$$v_z = k_{P2,z} e_{\dot{z}} \tag{10f}$$

Given the symmetry of the quadrotor system, the controller gains related to the x and y positions are set such that $k_{P1,x} = k_{P1,y} = k_{P1,xy}$, $k_{P2,x} = k_{P2,y} = k_{P2,xy}$, $k_{D,x} = k_{D,y} = k_{D,xy}$.

2.2.2. Inner-Loop

The inner-loop control law is designed by enhancing with feedback linearization the linear attitude controller presented in the Mathworks documentation regarding the UAV Toolbox Support Package for PX4 Autopilots [20]. The feedback linearization strategy is used to exactly linearize the quadrotor nonlinear dynamics and, thereby, to guarantee stability away from the equilibrium points. The following implementation exploits the Euler–Lagrange modeling formulation, which, in its correct form, can be used interchangeably with the N–E mathematical model, as shown in [16],

$$M' = W^{-T}[Bv + C\dot{\eta}] \tag{11}$$

where B and C are two 3×3 matrices defined as in [21], and where $v = \left[v_\varphi, v_\theta, v_\psi\right]^\top$ is a virtual control signal resulting from the inner-loop PD controller presented below. Compared to the one presented in [20], the inner-loop PD controller is designed using Euler rates instead of angular velocities, and it is formulated as follows:

$$e_{\dot{\varphi}} = k_{P1,\varphi}(\varphi_r - \varphi) - \dot{\varphi} \tag{12a}$$

$$e_{\dot{\theta}} = k_{P1,\theta}(\theta_r - \theta) - \dot{\theta} \tag{12b}$$

$$e_{\dot{\psi}} = k_{P1,\psi}(\psi_r - \psi) - \dot{\psi} \tag{12c}$$

$$v_\varphi = k_{P2,\varphi} e_{\dot{\varphi}} + k_{D,\varphi} \dot{e}_{\dot{\varphi}} \tag{12d}$$

$$v_\theta = k_{P2,\theta} e_{\dot{\theta}} + k_{D,\theta} \dot{e}_{\dot{\theta}} \tag{12e}$$

$$v_\psi = k_{P2,\psi} e_{\dot{\psi}} \tag{12f}$$

where $k_{P1,\varphi}$, $k_{P1,\theta}$, $k_{P1,\psi}$, $k_{P2,\varphi}$, $k_{P2,\theta}$, $k_{P2,\psi}$, $k_{D,\varphi}$, and $k_{D,\theta}$ are controller gains to be tuned. Given the symmetry of the quadrotor system, the controller gains related to the roll and pitch angles are set such that $k_{P1,\varphi} = k_{P1,\theta} = k_{P1,\varphi\theta}$, $k_{P2,\varphi} = k_{P2,\theta} = k_{P2,\varphi\theta}$, $k_{D,\varphi} = k_{D,\theta} = k_{D,\varphi\theta}$.

2.3. Reinforcement Learning

RL centers on training an agent that decides about taking actions by maximizing a long-term benefit through trial and error. RL is generally described by a Markov Decision Process (MDP). The agent–environment interaction in an MDP is illustrated in Figure 2. Agent, environment, and action represent the controller, controlled system, and control signal in engineering terms, respectively [22].

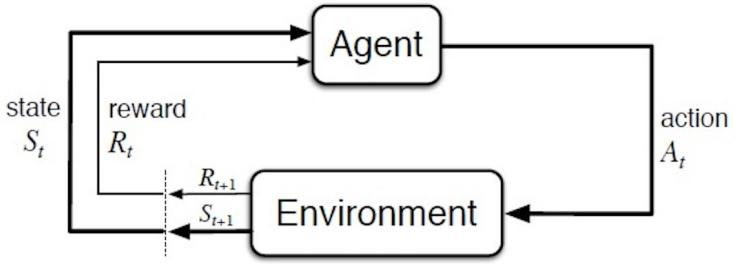


Figure 2. Interaction scheme between the agent and environment [22].

A DDPG algorithm is adopted, and it is designed to handle high-dimensional action spaces. It is an off-policy actor–critic algorithm that works based on the expected gradient of the action-value function that is given by

$$q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \tag{13}$$

where $q_\pi(s,a)$ denotes the action-value function for policy π at state s and action a; where $\mathbb{E}_\pi[\cdot]$ represents the expected value under policy π; where $\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ is the sum of discounted future rewards starting from time t in state s and represents the expected discounted return; and where γ is the discount rate, $0 \leq \gamma \leq 1$ [12,22,23].

The DDPG algorithm finds a deterministic target policy using an exploratory behavior policy. Thus, it outputs a specific action rather than a probability distribution over actions.

The actor–critic approach includes both a value function-based and a policy search-based method. While the actor refers to the policy search-based method and chooses actions in the environment, the critic refers to the value function-based method and evaluates the actor using the value function.

3. Proposed RL-Based Fine-Tuning

The main contribution of this work is the introduction, along with the first experimental results, of an RL policy to dynamically adapt control gains during flight. In fact, when considering a 'traditional' PD attitude controller, the main limitation is that the control gains, which have been set during the offline tuning phase, remain unchanged throughout the outdoor flight. By enhancing the controller with an adaptive law, such as the one that is here presented, the control gain can be fine-tuned during the flight to account for unmodeled dynamics and parameter uncertainties.

Given the online fine-tuning nature of the proposed RL method, a baseline controller tuning is performed as a first step. It is essential to properly adjust the controller parameters, as this directly impacts performance. Given a PD controller, an excessively high proportional gain may lead to overshooting, and a very high derivative gain may lead to instabilities. Thus, achieving a balance among controller gains is crucial to ensure a PD controller's accurate and stable response.

The MATLAB/Simulink 2024a environment was used for the presented simulation studies. While Simulink offers numerical automatic tools for PD controller tuning, this approach is only effective for linear plants or locally linearized plants. Consequently, manual tuning of PD parameters is performed, which consists in adjusting the control gains to minimize position and attitude error while avoiding excessive oscillations. Considering the authors' experience in UAV controller tuning, and as shown from the satisfactory results in Section 5, no alternative automatic tuning method (such as genetic algorithm) is deemed necessary at this stage.

Next, the RL-based controller parameter fine-tuning is implemented, which is illustrated as a block diagram configuration in Figure 3. The overall system consists of three components: controller, plant (the quadrotor in Figure 1), and agent (the RL component). The configuration of Figure 3 includes a trajectory planner, the linear transformation, and the parameter tuning parts. MATLAB/Simulink 2024a is utilized to train the agent for the RL-based fine-tuning of the inner-loop (attitude) controller. The notation has already been explained.

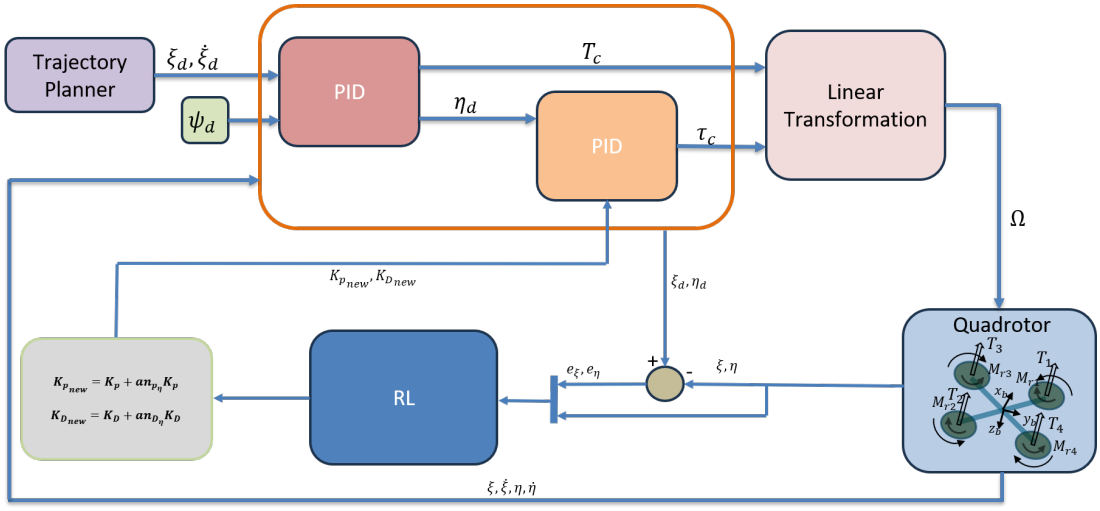


Figure 3. Block diagram of PD parameter tuning based on RL.

The state space S (for tuning) includes the positions (p) along the x, y, and z axes, as well as the Euler angles (η). It also includes the position errors (e_p) along the x, y, and z axes and the errors on the Euler angles (e_η). The agent learns to adjust dynamically the normalized weights ($n_{P_{\psi 2}}, n_{P_{\varphi\theta 1}}, n_{P_{\psi 1}}, n_{P_{\varphi\theta 0}}, n_{D_{\varphi\theta 0}}$), which creates the action space ($A \in \mathbb{R}^5$)

within the range of $[-1, 1]$ for all the controller parameters. The state and action spaces are denoted as follows:

$$\begin{aligned} S &= [p, \eta, e_p, e_\eta] \in \mathbb{R}^{12} \\ A &= [n_{P1,\varphi\theta}, n_{P1,\psi}, n_{P1,\varphi\theta}, n_{P2,\psi}, n_{D,\varphi\theta}] \in \mathbb{R}^5 \end{aligned} \tag{14}$$

Considering the normalized weights, the corresponding parameter tuning equations, related to the gains in (12a–f), are

$$\begin{cases} k_{P1,\varphi\theta,new} = k_{P1,\varphi\theta}(1 + an_{P1,\varphi\theta}) \\ k_{P1,\psi,new} = k_{P1,\psi}(1 + an_{P1,\psi}) \\ k_{P2,\varphi\theta,new} = k_{P2,\varphi\theta}(1 + an_{P2,\varphi\theta}) \\ k_{P2,\psi,new} = k_{P2,\psi}(1 + an_{P2,\psi}) \\ k_{D,\varphi\theta,new} = k_{D,\varphi\theta}(1 + an_{D,\varphi\theta}) \end{cases} \tag{15}$$

where a represents the search rate, and where $k_{P1,\varphi\theta,new}$, $k_{P1,\psi,new}$, $k_{P2,\varphi\theta,new}$, $k_{P2,\psi,new}$, and $k_{D,\varphi\theta,new}$ are the new control parameters fine-tuned by the trained RL agent. The manually tuned inner-loop controller parameters serve as the starting gain values for the training, while a combination of the search rate and normalized weights contribute to the tuning within an assigned interval relative to the initial controller gains.

The reward function is defined as a piece-wise function of the attitude error norm,

$$R(||e_\eta||) = \begin{cases} r_1, & \alpha_1 \leq ||e_\eta|| \\ r_2, & \alpha_2 \leq ||e_\eta|| < \alpha_1 \\ r_3, & \alpha_3 \leq ||e_\eta|| < \alpha_2 \\ r_4, & \alpha_4 \leq ||e_\eta|| < \alpha_3 \\ r_5, & \alpha_5 < ||e_\eta|| < \alpha_4 \\ r_6, & ||e_\eta|| \leq \alpha_5 \end{cases} \tag{16}$$

where r_i $(i = 1, \ldots, 6)$ are the reward values, and where α_i $(i = 1, \ldots, 5)$ represents the thresholds for the norm of the attitude errors, $||e_\eta||$. These thresholds define the conditions under which each reward value, r_i, is assigned.

The actor and critic neural networks (NNs) are designed as feed-forward NNs, the structures of which are depicted in Figures 4 and 5, respectively. The *tanh* activation function is chosen for the actor's output, giving the agent's output the value from the probability distribution between $[-1, 1]$.

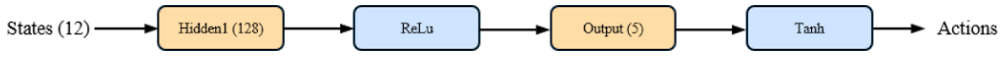


Figure 4. Structure of the actor neural network used for PD parameter tuning.

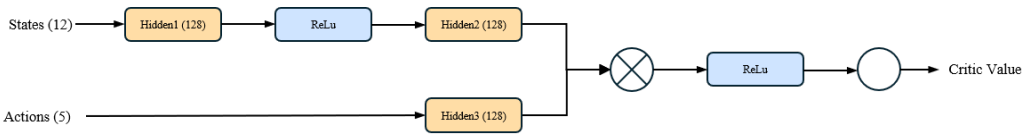


Figure 5. Structure of the critic neural network used for PD parameter tuning.

4. Agent Environment Setup for Controller Testing

The selected target hardware is the Pixhawk 2.1 Cube Black (ProfiCNC, Black Hill, Victoria, Australia) flight control unit, which supports PX4-Autopilot firmware. The controller is developed in MATLAB/Simulink 2024a, and the deployment process involves

overwriting the PX4-Autopilot flight controller v1.14.0 with the one resulting from the C++ code generation of the Simulink model. However, challenges arise when certain blocks in the Simulink model are incompatible with the code generation process, as they may not be supported. This limitation can prevent successful code generation, necessitating the modification or replacement of unsupported blocks to achieve a functional executable.

For the proposed controller, the RL agent Simulink block in Figure 6 is not supported for code generation.

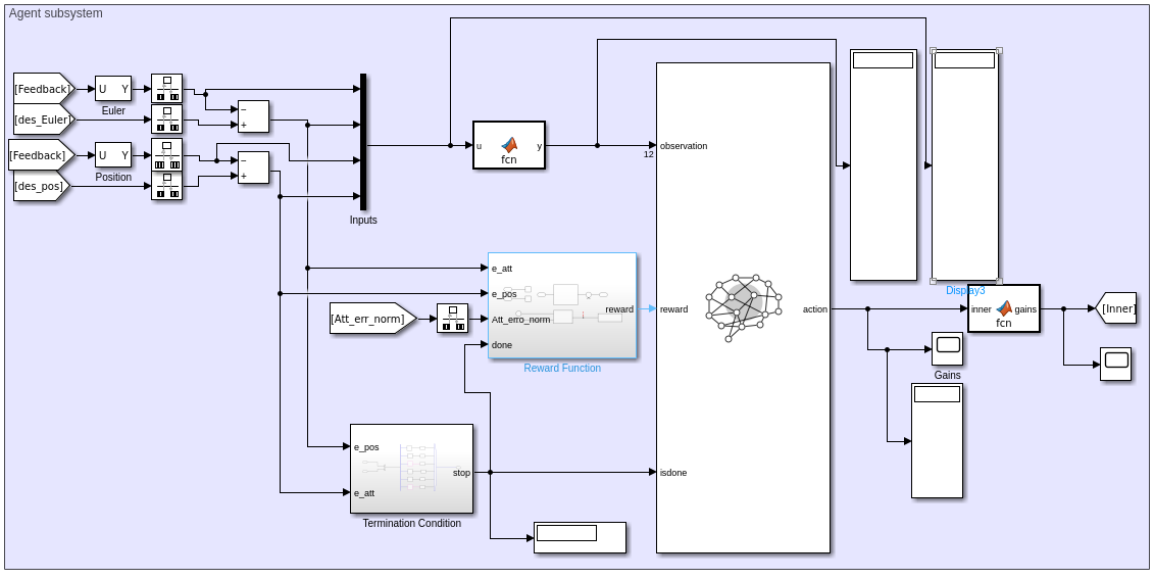


Figure 6. Reinforcement learning fine-tuning agent subsystem.

To overcome these limitations, an alternative approach is implemented: the weights and biases are extracted from the trained action model and the NN is reconstructed within the Simulink workspace. These trained weights and biases, represented as large matrices, are used to replicate the hidden layers of the action model. By applying the same series of mathematical operations of the hidden layers, and by utilizing the same observation function employed during the training phase, the original action can be identically reconstructed for the code generation.

In a reinforcement learning agent, the biases and weights within the neural network are fundamental components that allow the model to approximate intricate functions, such as policies or value functions. These parameters are iteratively updated during the training process to optimize the network's performance. The adjustments aim to reduce prediction errors while improving the agent's ability to maximize cumulative rewards, enabling it to learn effective strategies for decision-making in complex environments. By learning to minimize the position and attitude error, we expect the RL agent to approximate a control parameters nonlinear adaptation law to effectively compensate for discrepancies between the simulated and real external environment (such as sensor delays, unmodeled disturbances, and parameter uncertainties), consecutively increasing robustness. To this end, the neural network must have a sufficient number of neurons; however, in order to deploy the RL agent directly on the Pixhawk flight control unit, the flash memory limitations of the target hardware need to be taken into account. To balance performance and memory constraints, a neural network configuration of 128×128 neurons is selected. This setup provides satisfactory results while ensuring feasible memory utilization on the hardware.

Reconstruction of the Action Layer

The activation function *tanh* is applied to the output of the hidden layers in the network, as represented by the following equation:

$$hiddenLayerOutput = tanh(weightsLayer * observation + biasesLayer) \quad (17)$$

where *weightsLayer* and *biasesLayer* are the parameters specific to this layer, as the equation operates directly on the input observation. These parameters are responsible for determining the transformation of the input data before the nonlinear activation is applied.

For the final layer, the network uses a clipped ReLU function, defined as

$$action = min(N, max(Q, weightsLayer * hiddenLayerOutput + biasesLayer)) \quad (18)$$

where N and Q are the maximum and minimum allowable action values, respectively. These bounds are determined during the agent's training phase to ensure the output remains within a feasible range. In the case of RL-based fine-tuning of PD controller gains, the values of N and Q are set to 1 and -1, respectively, reflecting the permissible range for the controller's actions. This setup ensures that the network's outputs are appropriately scaled for the control task.

5. Results

In this research, the focus was on the problem of tracking a circular trajectory, also considering take-off and landing tasks. Since the RL agent was implemented for the attitude controller fine-tuning, only the attitude results are presented here, while the position results are listed in Appendix A for completeness. The trajectory under study was divided as follows: 10 s take-off, 2.5 s hovering, 20 s circumference (one lap), 2.5 s hovering, and 10 s landing for a total of 45 s. The real quadrotor's parameters were manually measured and the motors' characteristics were computed from the relative data-sheet. The resulting values are shown in Table 1, and they were used during the training and numerical simulations.

Table 1. Quadcopter measured parameters.

Parameter	Value	Unit Measurement
m_{tot}	1.2	kg
m_{prop}	0.01	kg
m_m	0.045	kg
m_{cg}	0.98	kg
r_{cg}	0.0625	m
h_{cg}	0.13	m
r_m	0.015	m
h_m	0.45	m
r	0.125	m
l	0.225	m
T	8.43	N
τ	0.1056	Nm
I_{xx}	0.0131	kg m^2
I_{yy}	0.0131	kg m^2
I_{zz}	0.0234	kg m^2
k_{motor}	1.4422×10^{-3}	$\frac{\text{kg m}}{\text{rad}^2}$
$\frac{c_D}{c_L}$	0.0237	-
b	3.1427×10^{-7}	$\frac{\text{kg m}}{\text{rad}^2}$

5.1. Agent Training Phase

As outlined in Section 3, before starting the RL training phase, the flight controller was manually tuned to achieve satisfactory tracking performance. After an iterative trial-and-error tuning process, the controller gain values in Table 2 were found and the resulting

numerical simulation attitude error norm was computed. The plot of the attitude error norm in Figure 7 highlights two main areas with higher spikes that correspond to the quadrotor's change of direction at 12.5 s and 32.5 s; hence, at the beginning and the end of the circular trajectory. In the hovering condition, the attitude was null; then, when starting the circular segment of the trajectory the attitude underwent a sudden change, which caused the spikes. The training of the agent aimed to reduce these high spikes in the attitude error norm.

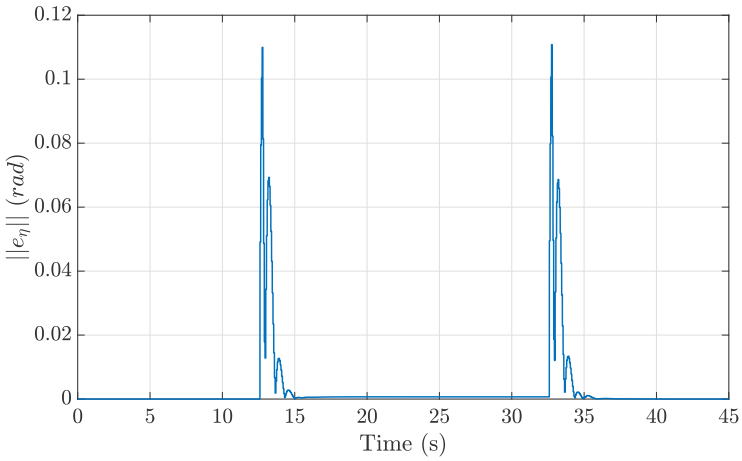


Figure 7. Attitude error norm of the manually tuned controller framework.

Table 2. Gains table.

Gain	Value	Name
$k_{P1,z}$	8.9	Position proportional gain on z-axis
$k_{P2,z}$	19.8	Velocity proportional gain on z-axis
$k_{P1,xy}$	0.6	Position proportional gain on xy-axis
$k_{P2,xy}$	3.9	Velocity proportional gain on xy-axis
$k_{D,xy}$	0.29	Velocity derivative gain on xy-axis
$k_{P1,\psi}$	2	Attitude proportional gain for yaw angle
$k_{P2,\psi}$	5.4801	Angular rate proportional gain for yaw angle
$k_{P1,\varphi,\theta}$	4	Attitude proportional gain for roll and pitch angles
$k_{P2,\varphi,\theta}$	11.467	Angular rate proportional gain for roll and pitch angles
$k_{D,\varphi,\theta}$	0.81905	Angular rate derivative gain for roll and pitch angles

According to the proposed methodology in Section 3, RL was then applied to further fine-tune the inner-loop controller gains. By heuristically assigning a value of $a = 0.4$ to (15), the following updating law was selected:

$$\begin{cases} k_{P1,\varphi\theta,new} = k_{P1,\varphi\theta}(1 + 0.4n_{P1,\varphi\theta}) \\ k_{P1,\psi,new} = k_{P1,\psi}(1 + 0.4n_{P1,\psi}) \\ k_{P2,\varphi\theta,new} = k_{P2,\varphi\theta}(1 + 0.4n_{P2,\varphi\theta}) \\ k_{P2,\psi,new} = k_{P2,\psi}(1 + 0.4n_{P2,\psi}) \\ k_{D,\varphi\theta,new} = k_{D,\varphi\theta}(1 + 0.4n_{D,\varphi\theta}) \end{cases} \tag{19}$$

In this work, the same reward function proposed in [15] was implemented, where the reward is defined as a function of the norm of the attitude error. Considering the simulation studies after the manually tuned controller gains, we observed that the norm of the attitude

error ranged from 10^{-4} to 1.1×10^{-1} rad, as shown in Figure 7. Given this relatively wide range, the reward function was set as the following piece-wise function of the attitude error:

$$R(||e_\eta||) = \begin{cases} -25, & 0.04 \leq ||e_\eta|| \\ -15, & 0.01 \leq ||e_\eta|| < 0.04 \\ -10, & 0.001 \leq ||e_\eta|| < 0.01 \\ -5, & 0.0005 \leq ||e_\eta|| < 0.001 \\ -1, & 0.0001 \leq ||e_\eta|| < 0.0005 \\ 10, & ||e_\eta|| \leq 0.0001 \end{cases} \tag{20}$$

The training phase was carried on in Matlab/Simulink 2024a, using the Deep Learning and Reinforcement Learning Toolbox; the simulation plant was modeled according to (2a–c); and the selected training hyperparameters were found heuristically and are displayed in Table 3. The sampling time for updating the control parameters was selected as a conservative rate that was one order of magnitude larger relative to the 5 ms attitude controller rate. Both of these sampling times were lower than the maximum admissible PWM signal rate of 2.5 ms of the adopted Pixhawk board and combination. The criterion for completion of training was based on achieving a target average reward, the value of which was determined by analyzing the distribution of the step counts across the error intervals shown in (20) relative to the norm of the attitude error observed with the manually tuned parameters.

Table 3. Hyperparameters.

Parameters	Value
Sampling time	0.05
Reward discount factor γ	0.99
Learning rate for actor	10^{-3}
Learning rate for critic	10^{-3}
L_2 regularization factor	10^{-5}
Optimizer parameter ϵ	10^{-8}
Minimum batch size	1024
Experience buffer length	10^6

According to (20) and the selected sampling time, the step count distribution yielded the following values: $[28, 24, 41, 350, 143, 345]$. The baseline reward value obtained from the piecewise function (20) was calculated as

$$-25 \cdot 28 - 15 \cdot 24 - 10 \cdot 41 - 5 \cdot 350 - 1 \cdot 143 + 10 \cdot 345 = 117.$$

This reward value serves as a reference point. To achieve better results, an improved reward value is required, necessitating adjustments to surpass this baseline value and drive enhanced performance in the training process.

5.2. Simulation Results

Once the training phase was completed, the RL agent-based controller was tested through numerical simulations. The attitude error norms for both the manually tuned controller and the RL-based fine-tuned controller are presented in Figure 8. The results demonstrate that the application of the fine-tuning method reduced overshoots and peak errors while enabling the system to reach steady-state conditions more quickly. This improvement is attributable to the RL agent, which optimized the controller by generating

a refined set of gains compared to the manually tuned values. These adjustments resulted in the enhanced performance and stability of the control framework. Table 4 presents the RMSE values for the attitude errors norm, providing a clear and quantitative comparison between the performance of the manually tuned controller and the RL fine-tuned one. A significant improvement in training performance is not expected in the simulation environment, as no external disturbances and unmodeled dynamics are present during the training and simulation phases.

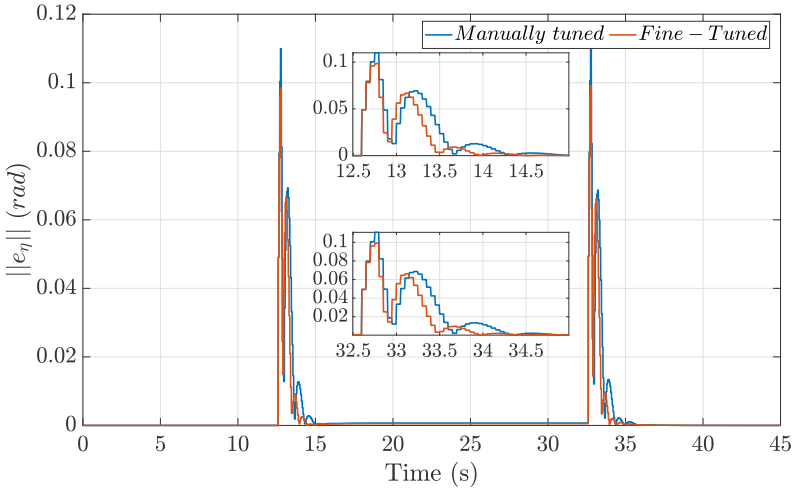


Figure 8. Attitude error norms of manually and RL-based fine-tuned controller frameworks.

Table 4. Numerical simulation RMSE of attitude error norm.

Numerical Simulation	Manually Tuned	Fine-Tuned
$\|e_\eta\|_{RMSE}$	12.75×10^{-3}	11.17×10^{-3}

Additionally, as shown in Figure 9, the RL agent set the fine-tuned controller gains differently from the initial manually tuned values right from the start of the simulation. Given the lack of disturbances in the numerical simulations, the fine-tuned gains remained constant and bounded throughout the simulation.

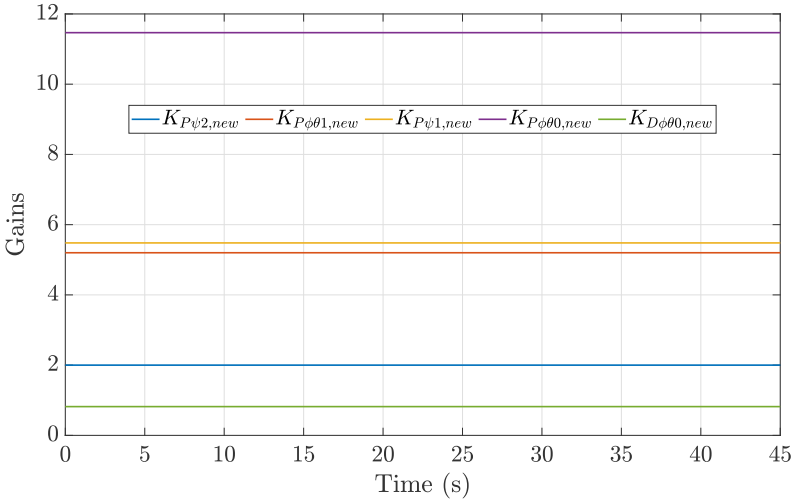


Figure 9. The inner-loop controller gains of the RL-based fine-tuned controller framework over time.

5.3. Experimental Results

The proposed methodology was deployed on quadrotor hardware to further test whether the RL agent-based controller performance improvements could be carried out in

real outdoor flight conditions. To this end, the C++ code was uploaded to the Pixhawk 2.1 Cube Black flight control unit, following the procedure in Section 4.

As in the previous section, the performances were compared, in terms of the norm of the attitude errors. Figure 10 compares the performance of the manually tuned gains with that of the fine-tuned gains, highlighting the differences in error magnitudes and response characteristics between the two approaches.

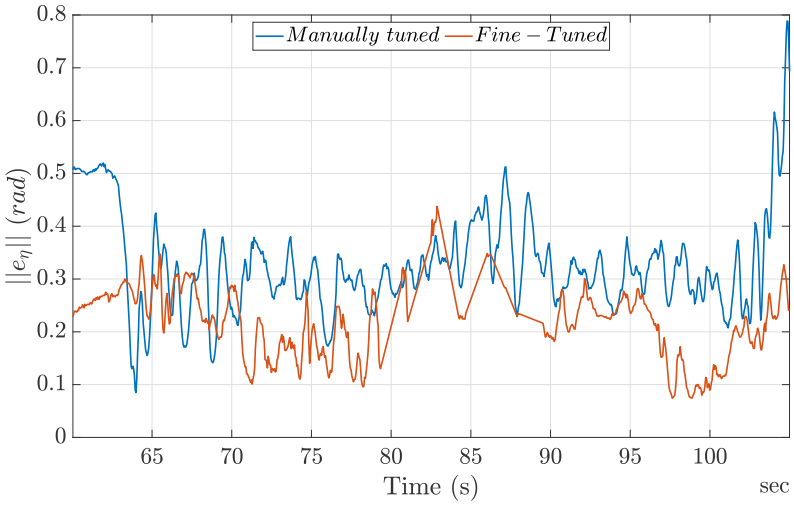


Figure 10. Attitude error norms of manually and RL-based fine-tuned controller frameworks for the experimental simulation.

Overall, the observed behavior for the trajectory showed noticeable performance enhancement, with the fine-tuned gains outperforming the manually tuned ones. This highlights the RL agent's ability to adapt and optimize control performance, even in the presence of real-world disturbances. These results can be further verified by comparing the respective RMSE values presented in Table 5. In the outdoor flight, since the RL agent learned to adapt the controller parameter, the performance improvement was more prevalent, at around 33%.

Table 5. RMSE of attitude error norm in the outdoor flight.

Outdoor Flight	Manually Tuned	Fine-Tuned
$\|e_\eta\|_{RMSE}$	33.93×10^{-2}	22.55×10^{-2}

As a further impressive result, the RL agent-based controller was able to adapt online to the controller gains. This behavior shows that, even if not necessary during numerical simulation ideal conditions, the agent had learned how to adapt the controller parameters in case of disturbances. The evolution of the RL-fine-tuned controller gains is shown in Figure 11, and, although initialized to match the manually tuned values, after approximately 2 s the gains began to adjust dynamically. These changes were moderate, avoiding extreme values, and they demonstrate the agent's ability to refine the control parameters in response to the system's requirements without overextending them. This behavior reflects a balanced adaptation aimed at maintaining system stability and performance.

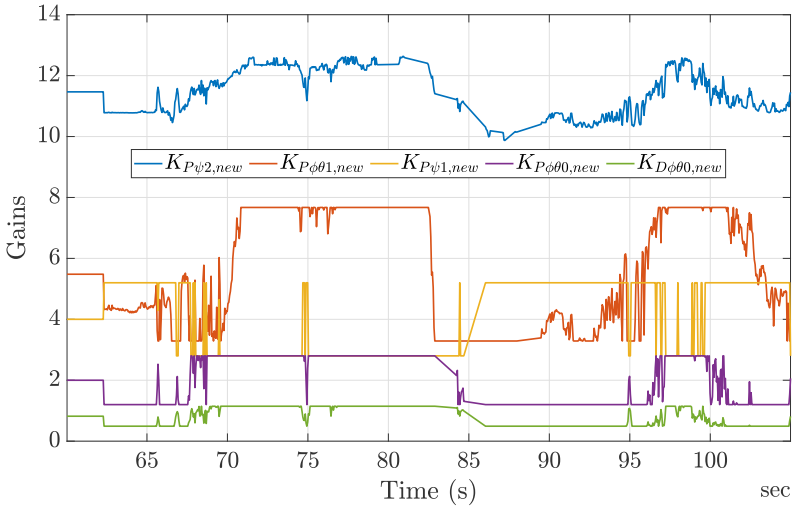


Figure 11. Time varying gains in outdoor experimental test results.

The agent operated without prior knowledge of the outdoor environment and had to contend with external factors, such as wind gusts and ground effects. Throughout the flight test, the agent's adaptive behavior was evident as it attempted to mitigate these disturbances. In some instances, the gains reached their maximum or minimum allowable values, reflecting the system's efforts to stabilize under challenging conditions. In addition, a few sudden spikes in the gains were observed, indicating rapid adjustments made by the agent to counteract abrupt changes in the environment. This behavior highlights the agent's responsiveness and the challenges posed by dynamic and unpredictable external forces. Overall, the application of the RL strategy improved the performances of flight outmatching those of the manually tuned gains.

6. Analysis

The results presented highlight both the potential and the challenges of applying reinforcement learning (RL)-based fine-tuning for PD controllers in real-world scenarios. The RL-based controller demonstrated improved tracking performance compared to the manually tuned PD controller, showcasing its ability to adapt and optimize controller parameters in challenging outdoor conditions.

The DDPG-based RL algorithm, an off-policy actor–critic approach, proved effective for fine-tuning the inner-loop gains of the controller. By leveraging this methodology, an adaptation law was achieved, capable of handling complex trajectories involving both circular paths and hovering segments, which mirrors realistic flight scenarios. It is worth noting that drag and gyroscopic effects were omitted during the training phase, but they were inevitably present in the outdoor flight conditions. Despite these simplifications during training, the RL-enhanced controller learned to adapt the PD control gains to external disturbances, and it consistently outperformed the manually tuned PD controller in the simulations and outdoor flight tests.

One of the key practical challenges in transferring an RL agent from simulation to a real-world system involves the deployment constraints of embedded hardware. In this study, the neural network architecture had to be designed to fit within the limited flash memory available on the Pixhawk flight controller. In doing so, the resulting memory requirements and computational complexity remained relatively constrained. Furthermore, the transition from RL simulation training to online agent deployment was partially enabled by the adopted development framework, which carried out RL training on the quadrotor dynamical model adopted in successful hardware in the loop (HIL) studies [17]. Although the training environment did not fully capture all real-world quadrotor dynam-

ics, a relatively detailed nonlinear model was used. A critical factor enabling successful sim-to-real transfer was the manual tuning of the PID controller on a model that incorporated PWM signal data type conversion (UINT16). Including this quantization effect proved essential, as it significantly influenced the resolution of the control inputs and the real-world behavior of the controller.

Ultimately, the findings of this study highlight the transformative potential of RL-based fine-tuning as a relatively computationally efficient solution for enhancing traditional UAV control strategies.

7. Conclusions and Future Works

This study successfully demonstrates the viability and effectiveness of a DDPG-based RL algorithm for fine-tuning PD controller parameters. The proposed method achieved substantial performance improvements in various test environments, including realistic outdoor scenarios. The RL-tuned controller exhibited enhanced adaptability, precise trajectory tracking, and robustness in the face of environmental disturbances. These results underscore the potential of RL-based approaches to bridge the gap between simulation and real-world applications in UAV control.

This study represents the initial stage in a broader research effort aimed at deploying reinforcement learning (RL)-based controllers on real-world quadrotors. While the present work focuses on the online fine-tuning of a PD controller using a DDPG-based RL agent and demonstrates successful implementation on real hardware, several important research directions remain open.

One of the key next steps is to conduct a formal robustness-and-stability analysis of the RL-tuned control system. The RL agent adapts controller gains online in response to changing conditions; however, ensuring consistent stability given real-world uncertainties—such as parameter variations, sensor noise, and external disturbances—requires theoretical validation. Future work will explore Lyapunov-based or other formal approaches to assess the closed-loop stability and robustness of the RL-based control system.

Additional future work will involve real-world deployment and evaluation of the presented approach to several UAV controllers, some of which have partially been explored in the authors' previous simulation-based works (such as RL-based PID, LQR, MRAC, and Koopman-integrated RL-based PID controllers), assessing their adaptability, performance, and practical challenges in experimental flight tests.

The current benchmark is limited to a comparison with a manually tuned PD controller on a single circular trajectory. To broaden the context, future studies will compare the RL-based method with other adaptive and optimal control strategies, such as MRAC, fuzzy-PID, genetic algorithm-based tuning, and Model Predictive Control (MPC) along several realistic trajectories. This will help quantify the performance trade-offs, training requirements, and applicability of each approach under varying conditions.

The current reward function uses a piecewise-constant structure based on attitude error thresholds, as adopted from the prior work [15]. While effective for initial implementation, this design may limit learning efficiency, due to discontinuous gradients. Future work will explore continuous reward shaping strategies that can promote smoother convergence and more optimal policy learning.

While the training model includes nonlinear dynamics and realistic features, such as actuator quantization (e.g., PWM type conversion), it omits certain aerodynamic effects like air resistance and gyroscopic torques. Future research will investigate the influence of these factors and whether including them in simulation improves real-world transferability. Further analysis will also examine how the RL agent compensates for these unmodeled dynamics during flight.

The framework developed in this paper has been validated on a quadrotor platform. We aim to evaluate its generalizability by applying it to additional aerial vehicles, including fixed-wing UAVs and other multirotor configurations, or unmanned surface vehicles (USVs), thereby testing the scalability and platform-agnostic nature of the proposed methodology.

Author Contributions: S.S., L.M., and S.M. were equally the main contributors to this work, with M.J.R., A.R., M.S., and K.P.V. reading, editing, and recommending additions and modifications to the contents. All authors have read and agreed to the published version of the manuscript.

Funding: S.S. has been partially supported by the Ministry of National Education of the Republic of Turkey on behalf of the Istanbul Medeniyet University, Turkey.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RL	reinforcement learning
DDPG	Deep Deterministic Policy Gradient
UAV	unmanned aerial vehicle
VTOL	vertical take-off and landing
ML	machine learning
DL	deep learning
N-E	Newton–Euler
ESC	electronic speed controller
MDP	Markov Decision Process
HIL	hardware-in-the-loop
GPS	global positioning system
RTK	real-time kinematic

Appendix A

In this appendix, the position error norm results of the simulations and the outdoor flights are presented for completeness.

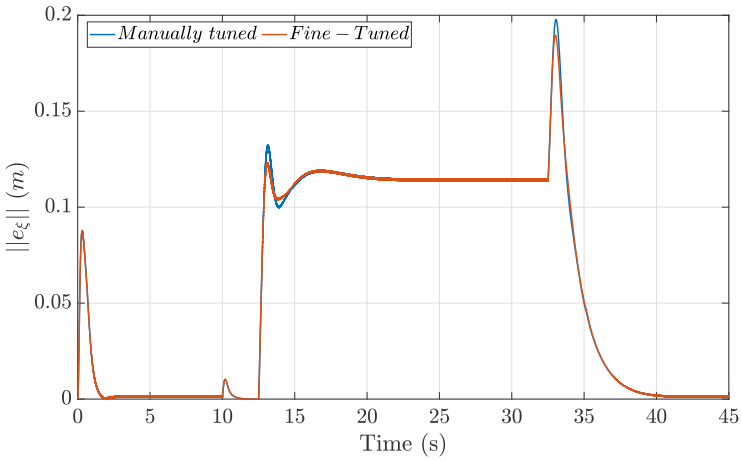


Figure A1. Position error norms of manually and RL-based fine-tuned controller in simulation.

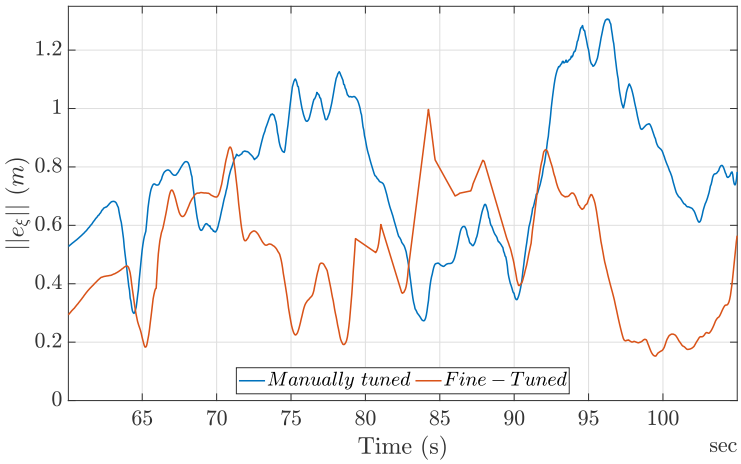


Figure A2. Position error norms of manually and RL-based fine-tuned controller in outdoor flight.

Table A1. Position error norm RMSE results comparison of the different simulations.

	Manually Tuned	**Fine-Tuned**
Simulation (m)	8.26×10^{-2}	8.24×10^{-2}
Outdoor (m)	80.59×10^{-2}	50.63×10^{-2}

References

1. Martinez, C.; Sampedro, C.; Chauhan, A.; Campoy, P. Towards autonomous detection and tracking of electric towers for aerial power line inspection. In Proceedings of the 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–30 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 284–295.
2. Ren, H.; Zhao, Y.; Xiao, W.; Hu, Z. A review of UAV monitoring in mining areas: Current status and future perspectives. *Int. J. Coal Sci. Technol.* **2019**, *6*, 320–333. [CrossRef]
3. Olivares-Mendez, M.A.; Fu, C.; Ludivig, P.; Bissyandé, T.F.; Kannan, S.; Zurad, M.; Annaiyan, A.; Voos, H.; Campoy, P. Towards an autonomous vision-based unmanned aerial system against wildlife poachers. *Sensors* **2015**, *15*, 31362–31391. [CrossRef] [PubMed]
4. Bassoli, R.; Sacchi, C.; Granelli, F.; Ashkenazi, I. A virtualized border control system based on UAVs: Design and energy efficiency considerations. In Proceedings of the 2019 IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–11.
5. Carrio, A.; Pestana, J.; Sanchez-Lopez, J.L.; Suarez-Fernandez, R.; Campoy, P.; Tendero, R.; García-De-Viedma, M.; González-Rodrigo, B.; Bonatti, J.; Rejas-Ayuga, J.G.; et al. UBRISTES: UAV-based building rehabilitation with visible and thermal infrared remote sensing. In Proceedings of the Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Lisbon, Portugal, 19–21 November 2015; Springer: Cham, Switzerland, 2016; Volume 1, pp. 245–256.
6. Li, L.; Fan, Y.; Huang, X.; Tian, L. Real-time UAV weed scout for selective weed control by adaptive robust control and machine learning algorithm. In Proceedings of the 2016 ASABE Annual International Meeting, Orlando, FL, USA, 17–20 July 2016; p. 1.
7. Carrio, A.; Sampedro, C.; Rodriguez-Ramos, A.; Campoy, P. A review of deep learning methods and applications for unmanned aerial vehicles. *J. Sens.* **2017**, *2017*, 3296874. [CrossRef]
8. Polydoros, A.S.; Nalpantidis, L. Survey of model-based reinforcement learning: Applications on robotics. *J. Intell. Robot. Syst.* **2017**, *86*, 153–173. [CrossRef]
9. Choi, S.Y.; Cha, D. Unmanned aerial vehicles using machine learning for autonomous flight; state-of-the-art. *Adv. Robot.* **2019**, *33*, 265–277. [CrossRef]
10. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; et al. Drone deep reinforcement learning: A review. *Electronics* **2021**, *10*, 999. [CrossRef]
11. Brunke, L.; Greeff, M.; Hall, A.W.; Yuan, Z.; Zhou, S.; Panerati, J.; Schoellig, A.P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control. Robot. Auton. Syst.* **2022**, *5*, 411–444. [CrossRef]
12. Sönmez, S.; Rutherford, M.J.; Valavanis, K.P. A Survey of Offline-and Online-Learning-Based Algorithms for Multirotor Uavs. *Drones* **2024**, *8*, 116. [CrossRef]
13. Yoo, J.; Jang, D.; Kim, H.J.; Johansson, K.H. Hybrid reinforcement learning control for a micro quadrotor flight. *IEEE Control Syst. Lett.* **2020**, *5*, 505–510. [CrossRef]

14. Mosweu, E.; Seokolo, T.B.; Akano, T.T.; Motsamai, O.S. Implementation of partially tuned PD controllers of a multirotor UAV using deep deterministic policy gradient. *J. Electr. Syst. Inf. Technol.* **2024**, *11*, 28. [CrossRef]
15. Sonmez, S.; Martini, S.; Rutherford, M.J.; Valavanis, K.P. Reinforcement Learning Based PID Parameter Tuning and Estimation for Multirotor UAVs. In Proceedings of the 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Chania, Greece, 4–7 June 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1224–1231.
16. Martini, S.; Valavanis, K.P.; Stefanovic, M.; Rutherford, M.J.; Rizzo, A. Correction to the Euler Lagrange Multirotor Model with Euler Angles Generalized Coordinates. *J. Intell. Robot. Syst.* **2024**, *110*, 17. [CrossRef]
17. Martini, S.; Mennea, S.M.; Mihalkov, M.; Rizzo, A.; Valavanis, K.; Sorniotti, A.; Montanaro, U. Design and HIL Testing of Enhanced MRAC Algorithms to Improve Tracking Performance of LQ-strategies for Quadrotor UAVs. In Proceedings of the 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE), Bari, Italy, 28 August–1 September 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 4206–4211.
18. L'afflitto, A.; Anderson, R.B.; Mohammadi, K. An introduction to nonlinear robust control for unmanned quadrotor aircraft: How to design control algorithms for quadrotors using sliding mode control and adaptive control techniques [focus on education]. *IEEE Control Syst. Mag.* **2018**, *38*, 102–121. [CrossRef]
19. Antonio, D.D. Controllo di Droni Multirotore Tramite Approccio Hardware-in-the-Loop. Master's Thesis, Università Politecnica delle Marche, Ancona, Italy, 2022.
20. MathWorks. Flight Visualization in Hardware-in-the-Loop (HITL) Simulation. 2025. Available online: https://www.mathworks.com/help/uav/px4/ref/flight-visualization-hitl-simulink.html (accessed on 24 March 2025).
21. Martini, S.; Sönmez, S.; Rizzo, A.; Stefanovic, M.; Rutherford, M.J.; Valavanis, K.P. Euler-Lagrange modeling and control of quadrotor UAV with aerodynamic compensation. In Proceedings of the 2022 International Conference on Unmanned Aircraft Systems (ICUAS), Dubrovnik, Croatia, 21–24 June 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 369–377.
22. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
23. Bilgin, E. *Mastering Reinforcement Learning with Python: Build Next-Generation, Self-Learning Models Using Reinforcement Learning Techniques and Best Practices*; Packt Publishing Ltd.: Birmingham, UK, 2020.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.