Reinforcement Learning based Optimal Guidance for Landing the Variable Skew Quad Plane on a Ship

Cansu Yıkılmaz*, and Christophe De Wagter†
Delft University of Technology, Delft, The Netherlands

Abstract

In this work, we present a reinforcement-learning-based approach for optimal guidance in landing a Variable Skew Quad Plane (VSQP) on a moving ship platform. We develop a reinforcement learning framework that computes the optimal acceleration inputs for the inner adaptive incremental nonlinear dynamic inversion-based controller. Through simulations, we evaluate the performance of various reward function combinations based on the key metrics: touchdown velocity, deviation, and duration. With the optimal reward for the given landing problem, we validate the approach on real ship data. Our results indicate that the reinforcement-learning-based approach outperforms the benchmark linear controller in achieving smoother and safer landings even under complex motion characteristics.

1 Introduction

Unmanned Aerial Vehicles have found use in a wide variety of applications, including environmental monitoring, search and rescue missions, and surveillance [1, 2] thanks to their capability to cost-effectively collect real-time data while being able to operate in remote and hazardous locations. And while sea environments constitute a large part of the planet, operations are often limited since the recovery of the UAV on the ship requires advanced autonomous landing technology.

The complexity of autonomous landing is a challenging, high-performance task. A big part of the challenge comes from the unpredictable motion under the effect of external disturbances and uncertainties [3]. In the case of ship landings, the unpredictability of the sea environment adds an extra layer of difficulty.

An autonomous landing problem has been extensively studied [5, 6, 7, 8, 9, 3] in terms of its sub-phases: target detection, relative state estimation, tracking and landing. The majority of studies rely on vision-based systems for target detection and relative state estimation tasks, with methods differing according to the characteristics of the landing platform (static vs dynamic). Vision-based systems are appreciated in maritime operations [10, 11] since reliance on inertial navigation alone is not feasible.

Concerning control, autonomous landing has relied on a variety of methods, ranging from PID controllers [8, 12, 13], adaptive control techniques [14, 15], robust flight [16] to model predictive control [17, 18] methods. While these techniques are effective in certain cases, they come with their own set of challenges. Classical feedback controllers often face problems with robustness due to dependency on fixed gains and perform poorly under the effect of external disturbances [19]. More advanced controllers like model predictive and robust flight control techniques require a detailed model and obtaining a perfect representation of the real-world system is often challenging. Moreover, the full formulation of the landing as an optimal control problem is numerically heavy. This highlights the need for more efficient and robust solutions in autonomous landing.

Reinforcement learning as a machine learning algorithm has gained significant popularity in recent years and found use in solving guidance and control problems [20, 21]. While optimal control techniques often require complete knowledge of the system dynamics, reinforcement learning is able to learn from raw environmental data without having prior information on the system and adapt to unknown dynamics and unpredictable changes in the environment.

In this work, we use reinforcement learning, particularly Proximal Policy Optimization (PPO), to develop an optimal guidance policy for landing the Variable Skew Quad Plane (VSQP) on a moving platform (ship). We feed the optimal control inputs (accelerations) and feed them back into an adaptive incremental nonlinear dynamic inversion-based controller, which includes a Weighted Least Squares (WLS) based optimization routine for control allocation. We account for the motion in the z-direction and search for the best objective function to minimize landing impacts and lateral offset within a given time. Finally, we test our approach with real-world ship data and compare it with a PID controller.

2 Related Work

Reinforcement learning has become an effective approach for solving autonomous landing problems. One of the earliest studies by Polvara et al. [22] utilized Deep Q-Networks (DQN) for landing on a static platform using low-resolution images coming from a downward-facing camera. Later, Lee et al. [23] developed an actor-critic framework where they included a PID-based inner attitude controller, a ground-

*Email address(es): cnsyklmzoutlook.com
†Email address(es): c.dewagtertudelft.nl

(a) VSQP in quad-mode configuration. (b) VSQP in transition (c) VSQP in forward flight configuration.

Figure 1: The configuration modes of the VSQP: Variable Skew Quad Plane [4]

looking camera model, and a laser rangefinder to enable precise autonomous landing on a static target. The challenge of landing on a dynamic target was addressed by Rodriguez et al. in [24], where they used Deep Deterministic Policy Gradient (DDPG) with Gazeebo. Xie et al. [25] later divided the problem into two parts: perception and relative pose estimation on the one hand, and trajectory optimization and control on the other. In contrast to the previous studies, they also accounted for sensor noise, intermittent measurements, randomness in UAV movement, and incomplete or inaccurate observations. Moreover, they explored a hybrid approach that combined reinforcement learning with heuristic methods for tracking and landing tasks.

In the context of ship landing, Saj et al. [26] claim that previous studies lack the robustness component, which leads to the development of not fully efficient algorithms for ship landing problems. They adapted the twin delayed DDPG (TD3) algorithm for a vision-based ship landing. To overcome the reality gap problem, they applied a domain randomization approach [27] through a variation of environment parameters. They showcased the superior performance of their framework over a PID controller in the real-life tests made under varying wind conditions. These studies collectively highlight the capabilities of reinforcement learning in solving autonomous landing problems.

3 VARIABLE SKEW QUAD PLANE (VSQP)

The Variable Skew Quad Plane is a hybrid UAV designed by the Micro Air Vehicle Laboratory (MAVLab) at TU Delft. The VSQP has two modes of operation - hover and cruise. It is controlled through the skew angle Λ. In hover mode ($\Lambda = 0\,\mathrm{deg}$), the drone operates as a quadrotor with an extra pusher propeller. In the cruise mode ($\lambda = 90$), it operates as a fixed-wing aircraft and achieves forward speed with a push propeller located at the tail. The VSQP, with an intermediate skew angle, resembles an Oblique Flying Wing (OFW) plane. This design offers better gust rejection in hover mode as the wing hides against the fuselage and lower drag during cruise mode since the hover motors hide in the fuselage. The configuration modes of the VSQP are shown in Figure 1.

The guidance and control model of the VSQP incorporates an Adaptive Incremental Nonlinear Dynamic Inversion (ANDI) controller with the Weighted-Least-Square-based control allocation, as the VSQP is an overactuated system [28]. This controller is implemented as a "one-loop" architecture as opposed to cascaded controller approaches in previous ANDI work [29]. This allows the control to distribute the load optimally throughout the control surfaces of the VSQP. The diagram showing the overall guidance and control scheme of the VSQP is given in Figure 2. Within the $G\&C$ structure, the WLS block takes the virtual control parameters calculated by the error controllers for the position and attitude. The reference model is used to shape the given desired setpoint into reference signals that are feasible for the platform.

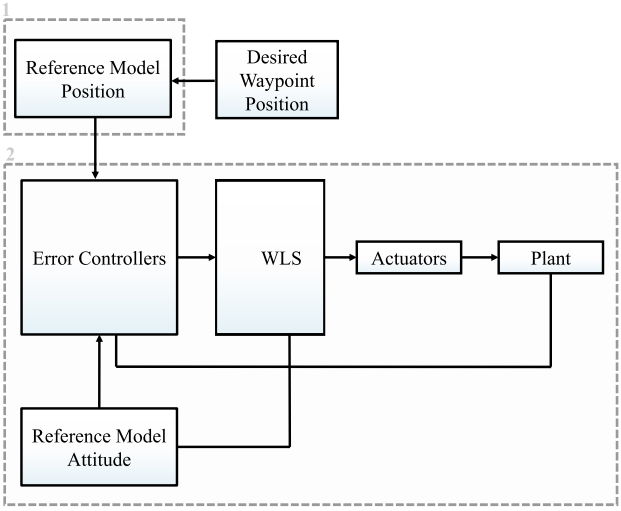

Figure 2: Adaptive Incremental Nonlinear Dynamic Inversion (ANDI) controller with WLS-based control allocation

To perform the optimal path planning, the guidance scheme of the VSQP is replaced by a reinforcement learning-based model to learn the optimal acceleration setpoints. The simplified diagram for the reinforcement learning framework is shown in Figure 4. The reinforcement learning framework contains a deep neural network structure and takes the states of the VSQP as inputs and generates the corresponding optimal acceleration inputs, which are denoted as a_{des}. The response of the vehicle with its ANDI controller, symbolized as

a_{real}, corresponds to the actual accelerations resulting from the applied control inputs. To effectively map the given commands to the actual accelerations and use them in the training process, a simplified model for both the UAV and the controller is necessary. While previous ANDI work [30] suggests a first-order transfer function for normal quadrotor dynamics, the optimization routine in the VSQP is more complex.

In addition to the existing actuators of the VSQP, the WLS also uses pitch and roll angles as its virtual control inputs within its optimization routine, which are later fed back to the attitude reference model. These Euler angles have the slowest response among all actuators, therefore limiting the vehicle's response to the given control input. A simplified model for the VSQP is therefore obtained by modeling the attitude response of the complete ANDI+VSQP. The result is a transfer function that maps the desired accelerations to the actual ones. It was mathematically derived from the Simulink model and is given in Eq. 1.

$$acc_{tf} = \frac{206s^3+861.1s^2+287s+95.68}{s^6+20.22s^5+143.3s^4+682s^3+1021s^2+340.2s+95.68} \quad (1)$$

The response of the transfer function to a step input is shown in Figure 3. To improve the numerical stability, an equivalent fourth-order model is used to avoid the numerical issues.

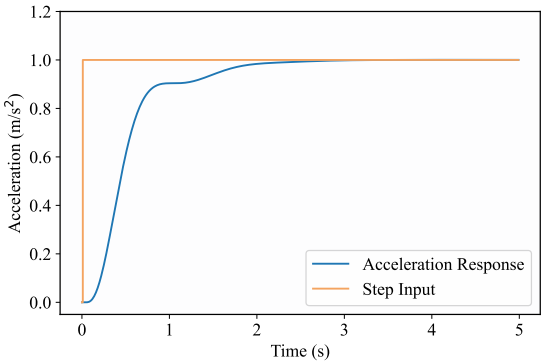

Figure 3: Response of the VSQP to a step input

4 Reinforcement Learning Based Problem Formulation

Reinforcement learning has an agent interact with an environment to take actions that maximize a cumulative reward over time. At each time step t, the agent receives a state s_t from a state space $\mathcal{S}$ and chooses an action a_t from an action space $\mathcal{A}$, based on a policy $\pi(a_t \mid s_t)$. The policy could be stochastic $\pi(a \mid s)$, with a probability associated with each possible action, or deterministic $\pi(s)$, indicating the agent's decision-making process from states to actions. The agent receives a reward r_t for its actions and transitions to the next state s_{t+1}, according to the reward function $\mathcal{R}(s,a)$ and state transition probability $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. This process continues until the agent reaches a terminal state and restarts. The cumulative reward is defined as the return function $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ discounted with a factor $\gamma \in (0,1]$.

The expectation of cumulative reward for a specific policy is represented as V^π, the value function, and is given as:

$$V^\pi(s_t) = \mathbb{E}\left[R_t \mid s_t, a_t = \pi(s_t)\right] \quad (2)$$

The action-value function Q^π in Eq. 3 is equivalent to the value function for every action-state pair (s_t, a_t).

$$Q^\pi(s_t,a_t) = r(s_t,a_t) + \gamma \sum p(s_{t+1} \mid s_t,a_t) V^\pi(s_{t+1}) \quad (3)$$

The goal is to find the optimal policy π^* as in Eq. 4 that maximizes the value function. The corresponding value function is called the optimal value function.

$$\pi^* = \arg\max_\pi V^\pi(s_t) \quad (4)$$

Policy gradient methods are a class of reinforcement learning algorithms that are particularly effective in environments with continuous state and action spaces. In a policy gradient algorithm, the policy is presented as π_θ where θ are neural network parameters. Given that the expected return is $J(\theta) = \mathbb{E}_s[V_{\pi_\theta}(s)]$, the goal is to find optimal parameters $\theta^* = \arg\max_\theta J(\theta)$ through a gradient ascent update $\theta_{k+1} = \theta_k + \alpha_k \nabla J(\theta_k)$ where α_k is the learning rate. The gradient of $J(\theta)$ is calculated using the policy gradient theorem, which is given by:

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}\left[Q_{\pi_\theta}(s,a) \nabla \log \pi_\theta(s,a)\right] \quad (5)$$

There are many types of policy gradient algorithms, such as TRPO [31], SAC [32], and PPO [33]. In this work, we adapt the PPO algorithm to solve the landing problem of the VSQP on a moving platform. PPO is a highly stable and efficient policy gradient algorithm that introduces techniques like clipping the objective and using an adaptive penalty coefficient to improve the stability and efficiency of the learning process [33]. It employs an actor-critic model where the actor takes output actions and the critic evaluates its performance.

PPO uses the following objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \quad (6)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio of the action under the new and old policies, $\hat{A}_t$ is advantage at timestep t, and ϵ is a hyperparameter that defines the clipping range to keep the ratio within reasonable bounds.

Problem Formulation

The UAV landing problem can be described as a Markov decision process (MDP). Typical MDP can be defined as a four tuple $(\mathcal{S}, \mathcal{A}, R, \Omega)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R(s,a)$ represents the immediate rewards, and

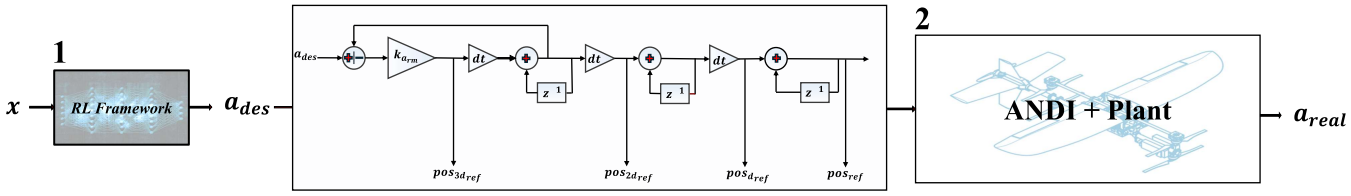

Figure 4: Reinforcement learning framework integrated into the existing control structure

Ω is the observation space. Based on this model, the landing problem is formulated as:

1. *State Space* $\mathbb{S}_d$: The state space $\mathbb{S}_d$ represents the states of the UAV. Give the 4th order transfer function model of ANDI+VSQP, the state space for the drone consists of extra three acceleration derivatives $\dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d$ on top of the position $\boldsymbol{p}_d$, velocity $\boldsymbol{v}_d$, and acceleration $\boldsymbol{a}_d$ components:

$$\mathbb{S} = \{\boldsymbol{p}_d, \boldsymbol{v}_d, \boldsymbol{a}_d, \dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d\} \tag{7}$$

2. *State Space* $\mathbb{S}_s$: The state space $\mathbb{S}_s$ represents the states of the ship platform. It consists of platform position p_s and velocity v_s components:

$$\mathbb{S}_s = \{\boldsymbol{p}_s, \boldsymbol{v}_s\} \tag{8}$$

The motion of the ship is characterized by sinusoidal signals that are governed by specific frequencies, amplitudes, and initial conditions. The definitions for position and velocity are defined as:

$$p_i(t) = A_{\mathrm{p},i} \sin(\omega_i t + \phi_i) + p_{\mathrm{init},i} + \xi_i(t) \tag{9}$$

$$v_i(t) = \omega_i A_{\mathrm{p},i} \cos(\omega_i t + \phi_i) + \eta_i(t) \tag{10}$$

where $A_{\mathrm{p},i}$ is the amplitude of the sinusoidal motion for each axis i, ω_i denotes the angular frequency, ϕ_i is the phase shift, and finally $\eta_i(t)$ captures the random walk component. This random component helps to simulate unpredictable movements or external disturbances affecting the ship's motion.

3. *Observation Space* Ω: The input to the reinforcement learning algorithm is given as the relative position p_r and velocity v_r components of the drone with respect to the moving platform, along with the *real acceleration* $\boldsymbol{a}_d$ of the drone:

$$\mathbb{S}_s = \{\boldsymbol{p}_r, \boldsymbol{v}_r, \boldsymbol{a}_d\} \tag{11}$$

4. *Action Space* $\mathbb{A}$: The action space Eq. 12 $\mathbb{A}$ consists of *desired acceleration* components $a_{n_{des}}, a_{e_{des}}, a_{d_{des}}$ determined based on the inputs of the reference model:

$$\mathbb{A} = \{a_{n_{des}}, a_{e_{des}}, a_{d_{des}}\} \tag{12}$$

Within the reinforcement learning framework, the control input parameters a_n and a_e are used to steer the drone onto the platform, while a_d is mainly used to find the right time to accomplish landing.

5. *Reward Function* R: Designing a reward function that is suitable for a *landing on a moving target* problem is not a straightforward task. Several options are defined and we classify the reward function in terms of three components: position tracking, velocity tracking, and collision penalty.

   (a) Position Tracking: The reward function R_p in Eq. 13 is designed for tracking of the moving platform by the minimization of the deviation in both vertical and horizontal planes relative to the target:

$$\begin{aligned} R_p = k_1 \left|p_{r_{\mathrm{old}_v}}\right| - k_1 \left|p_{r_{\mathrm{new}_v}}\right| \\ + k_2 \left\|\left\|\mathbf{p}_{r_{\mathrm{old}_h}}\right\| - k_2 \left\|\mathbf{p}_{r_{\mathrm{new}_h}}\right\|\right\| \end{aligned} \tag{13}$$

   The parameters $p_{r_{\mathrm{old}_v}}$ and $p_{r_{\mathrm{new}_v}}$ represent the old and new vertical distances; $\mathbf{p}_{r_{\mathrm{old}_h}}$ and $\mathbf{p}_{r_{\mathrm{new}_h}}$ represent the horizontal distances between the UAV and the target, respectively. The term $\left|p_{r_{\mathrm{old}_v}}\right| - \left|p_{r_{\mathrm{new}_v}}\right|$ rewards the agent based on the absolute difference in vertical position and $\|\mathbf{p}_{r_{\mathrm{new}_v}}\| - \|\mathbf{p}_{r_{\mathrm{new}_h}}\|$ uses the norm of both x and y components together to compute the difference. The relative importance is defined by the weights k_1 and k_2, which are determined to be 2 and 1, respectively.

   (b) Velocity Tracking: The velocity reward R_v is designed to ensure that the speed of the UAV is adjusted in a way that it slows down while it gets closer to the ship. Similarly to the position reward R_p, the parameters $v_{r_{\mathrm{old_v}}}$ and $v_{r_{\mathrm{old_h}}}$ represent

the old and new relative velocities in the vertical plane, whereas $\mathbf{v}_{r_{\text{old}_\text{v}}}$ and $\mathbf{v}_{r_{\text{old}_\text{h}}}$ represent the corresponding parameters in the horizontal plane. The reward function is dynamically scaled by a factor that depends on the drone's height above the ship.

$$R_v = k_1 \left(v_{r_{\text{old}_\text{v}}} - v_{r_{\text{new}_\text{v}}}\right) \left(\frac{h_t + h}{h_t}\right) + k_2 \left(v_{r_{\text{old}_\text{h}}} - v_{r_{\text{new}_\text{h}}}\right) \left(\frac{h_t + h}{h_t}\right) \quad (14)$$

(c) Collision Penalty: The collision penalty is designed to directly penalize the UAV's speed at the touchdown point. To assess the performance of the landing, two different reward functions are considered based on relative velocities and absolute velocities.

The first reward function R_{c_1} as given in Eq.15 is defined as the *relative velocity collision penalty* that penalizes the relative velocity of the UAV with respect to the platform. The constants k_3 and k_4 allow tuning the sensitivity of the penalty to the different directional velocities, avoiding lateral collisions as well as maintaining a safe descent speed.

$$R_{c_1} = k_3 \left|v_{r_{\text{new}_v}}\right|^2 + k_4 \left\|\mathbf{v}_{r_{\text{new}_h}}\right\|^2 \quad (15)$$

The second collision reward R_{c_2}, *absolute velocity collision penalty*, focuses on the vertical velocity components at the point of landing. The $v_{d_{\text{new}_z}}$ indicates the vertical velocity of the drone, and $v_{p_{\text{new}_z}}$ refers to the platform's vertical velocity.

$$R_{c_2} = k_5 \left|v_{d_{\text{new}_z}}\right|^2 + k_6 \left|v_{p_{\text{new}_z}}\right|^2 \quad (16)$$

These collision rewards only get active at the final point of landing. This approach resembles a form of Cliffwalk [34] since the agent does not receive intermediate rewards till the point of touchdown. While this makes the search for an optimal policy more challenging, the agent gains greater freedom in its actions as opposed to the reward functions used in the previous studies [24, 26, 25], which defined a separate reward function for each altitude range.

These reward functions are strategically combined to identify the most effective objective for the given problem. The table 1 outlines the combinations used in this study. Reward R1 only consists of the position reward R_p, encouraging the agent to minimize the difference between the old and new positions. R2 and R3 incorporate the collision penalties R_{c_1} and R_{c_2} along with the position reward. Lastly, rewards R4 and R5 combine everything, including the velocity reward R_v, with the collision penalties.

Table 1: Reward combinations used in the study

Reward Number	Combination
R1	R_p
R2	$\text{R}_p + \text{R}_{c_1}$
R3	$\text{R}_p + \text{R}_{c_2}$
R4	$\text{R}_p + \text{R}_v + \text{R}_{c_1}$
R5	$\text{R}_p + \text{R}_v + \text{R}_{c_2}$

5 TRAINING

For training the reinforcement learning framework, a gym environment including classes for both the UAV and the ship platform dynamics was created in Python.

The ship model uses sinusoidal signals with mixed frequencies and random walk components to simulate a wide variety of signals that resemble real ship motion. This variety in the ship motion is critical to prevent overfitting during the training process. In alignment with the observation space Ω from Eq.11, the relative position, relative velocity, and real drone accelerations are given to the model at every time step. The outputs are the desired accelerations, which are later used to propagate the states of the drone.

Since the ship is moving at a slow constant speed and this study focuses on relative landing control, the constant forward motion of the ship is ignored, and only the wave-induced hard-to-predict random motions are modeled. In the horizontal plane, the relative navigation of the UAV is evaluated by selecting different UAV initial conditions. The training process is repeated for all the combinations indicated in Table 1. The simulations are ended when any of the following conditions are met: a ground collision occurs, the drone exits the designated area, the maximum number of steps is reached, or the drone achieves a landing condition of being below 0.1m above the platform.

5.1 Initializing the drone dynamics

For training, a vectorized environment is used to run several environments in parallel. To do so, initial drone states are uniformly sampled from the following integrals:

$$\begin{array}{lll} p_n \in [-1,1] + p_{n_i} & p_e \in [-1,1] + p_{e_i} & p_d \in [-1,1] + p_{d_i} \\ v_n \in [-0.5, 0.5] & v_e \in [-0.5, 0.5] & v_d \in [-0.5, 0.5] \\ a_n \in [-0.05, 0.05] & a_e \in [-0.05, 0.05] & a_d \in [-0.05, 0.05] \end{array} \quad (17)$$

Here, $p_{n_i}, p_{e_i}, p_{d_i}$ represent the initial position elements of the drone. The parameters p_{n_i} and p_{e_i} are chosen in a way that the drone aligns itself with the platform and p_{d_i} is determined to be 10 meters as the starting altitude at the

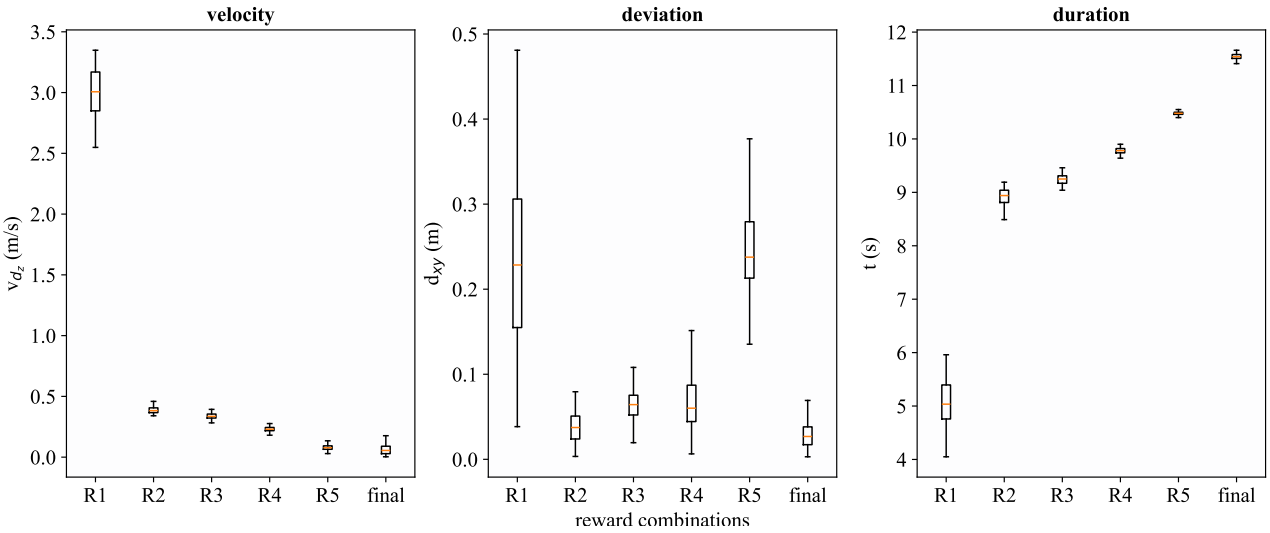

Figure 5: The analysis of reward combinations in terms of performance metrics

beginning of the training procedure. Additionally, nine additional state elements which are the derivatives of accelerations ($\dot{\boldsymbol{a}}_d, \ddot{\boldsymbol{a}}_d, \dddot{\boldsymbol{a}}_d$) up to a third degree are also initialized with a value of zero with the given state elements.

5.2 *Initializing the ship dynamics*

Similarly to the drone model, ship states are uniformly sampled from the intervals:

$$\begin{array}{lll} A_x \in [-0.1, 0.1] & A_y \in [-0.1, 0.1] & A_z \in [-0.1, 0.1] + A_z \\ f_x \in [-0.1, 0.1] & f_y \in [-0.1, 0.1] & f_z \in [-0.1, 0.1] + f_z \end{array} \quad (18)$$

For the ship, wave parameters amplitude A and frequency f are used to initialize the sinusoidal motion characteristics. To create variation in the generated sinusoidal motions additional parameters such as mix frequency f_{mix}, random walk step size rw_{ts}, and low pass filter coefficient α are also given as an input to the model. The values for these parameters are determined as 0.2, 0.01, and 0, respectively.

6 Evaluation

The performance of reward functions from Table 1 is assessed based on three key metrics: touchdown velocity, deviation, and duration. This evaluation is completed for a specific test case scenario defined as follows. A sinusoidal wave in the z-direction is generated using the characteristics:

$$A_z \in [-0.1, 0.1] + A_z, \quad f_z \in [-0.1, 0.1] + f_z \quad (19)$$

The touchdown velocity is a critical metric for assessing the safety and smoothness of the UAV's landing. Too high velocities result in UAV damage or even a crash. The deviation metric measures the accuracy of the UAV's landing by evaluating how close the UAV lands to the intended target point. Lastly, time to collision represents the duration that a landing takes.

The assessment of the drone is visualized with box plots in Figure 5. The left subplot shows the distribution of touchdown velocities for each reward combination. The y-axis represents the vertical descent velocity v_{d_z} in meters per second (m/s), and the x-axis lists the reward combinations (R1 to R5) and the final (R5 with tuned coefficients). R1 has the highest median touchdown velocity at around 3.0 m/s. In contrast, the value of the median velocity drops significantly for R2 and R3, while the latter has a slightly lower median value. This shows that having a collision penalty, either absolute or relative, has a significant effect in reducing touchdown velocity. With the addition of the velocity element in R4 and R5, the velocity values further decrease, with touchdown velocities close to zero, indicating the landing is extremely smooth and controlled.

The middle box plot shows the deviation from the target center (d_{xy}) in meters for each reward combination. While the deviation values are all within landing margins, for rewards R2, R3, and R4, the deviation is less, while R1 and R5 show that the drone deviates a bit more from the center. Finally, the right plot shows the landing duration in seconds. As expected, the duration gets longer as the velocity values decrease, resulting in a less aggressive flight.

While different performance parameters can be of importance, in practical flights, more importance is put on having lower velocities. Based on the results given in the top-left figure, the reward function R5, which uses the absolute velocities of the drone, has been identified as having the optimal form for the given problem.

Further analysis was conducted to determine the effect of coefficients (k_5 and k_6) on the UAV's performance, which was previously given a value of 1. Both coefficients k_5 and

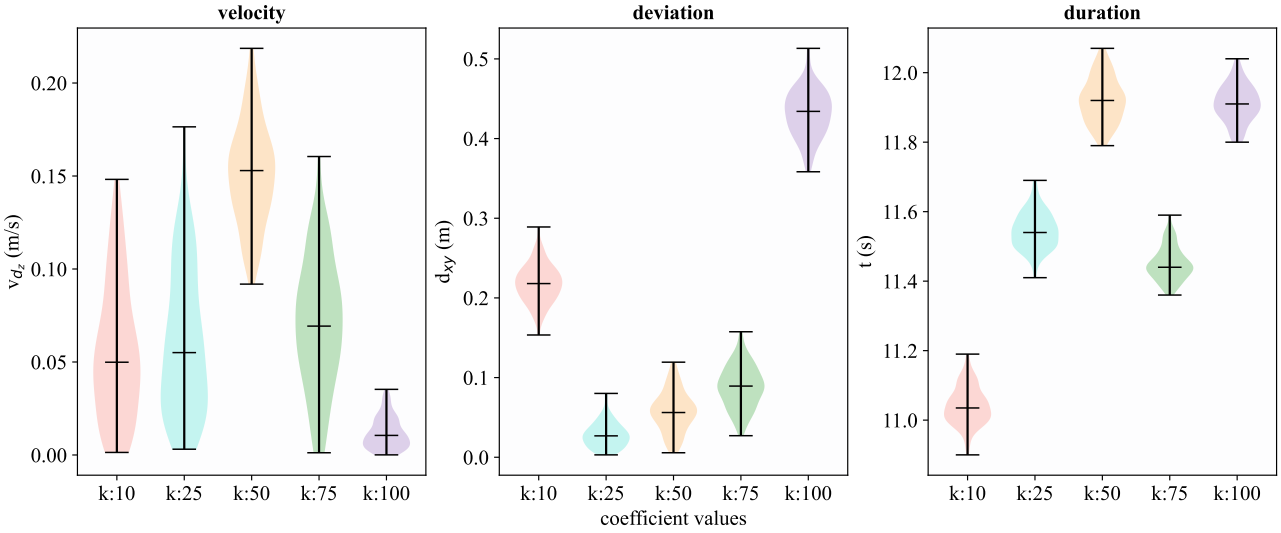

Figure 6: The analysis of reward combinations in terms of performance metrics

k_6 are tested with the values of 10, 25, 50, 75, and 100, respectively. The results are given in the Figure 6, which show the effect of the different coefficients on the UAV's performance, for the chosen reward combination.

The first plot on the left again shows the distribution of touchdown velocities, but this time for the same combination, R5. Among the coefficients, k:100 achieves the lowest median value of a mere 0.01 m/s. It was also observed that much higher values than 100 either lead to extremely long flights or no landing at all. Specifically, high penalty values for the collision make the drone extremely hesitant to take any action. These are left out of the plot as they were deemed unsuccessful.

The graphs indicate that coefficients between 10 and 100 provide a good balance in terms of performance metrics, but do not show a very linear trend. Considering the negative effect of higher penalties on performance and the low deviation values, the coefficient k:25 was chosen for R5. The final form of the reward function is thus given as:

$$\begin{aligned} R_{final} = & \left|p_{r_{\text{old}_v}}\right| - \left|p_{r_{\text{new}_v}}\right| + 2\left\|\mathbf{p}_{r_{\text{old}_h}}\right\| - 2\left\|\mathbf{p}_{r_{\text{new}_h}}\right\| \\ & + \left(v_{r_{\text{old}_v}} - v_{r_{\text{new}_v}}\right)\left(\frac{h_t + h}{h_t}\right) \\ & + 2\left(v_{r_{\text{old}_h}} - v_{r_{\text{new}_h}}\right)\left(\frac{h_t + h}{h_t}\right) \\ & + 25\left|v_{d_{\text{new}_z}}\right|^2 + 25\left|v_{p_{\text{new}_z}}\right|^2 \end{aligned} \tag{20}$$

7 Validation

To determine if the reinforcement framework could successfully land on waves, validation with real ship data is performed. Real ship data was gathered at a frequency of 5 Hz and interpolated using a quadratic fit to the 10 Hz used in previous simulations. A sample portion is shown in Figure 7.

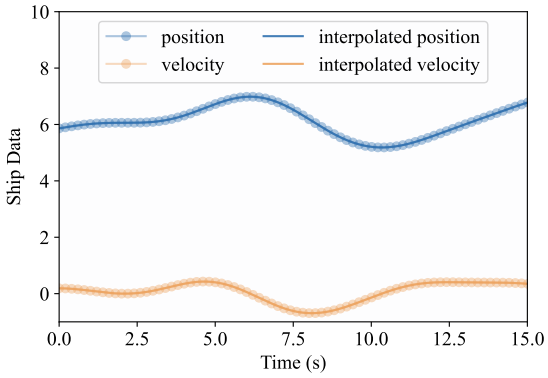

Figure 7: Measured ship motion data for validation

The performance of the reinforcement learning framework was tested on the real ship data using the final form of the reward function (see Eq. 20). Additionally, the RL framework was compared against two benchmark controllers.

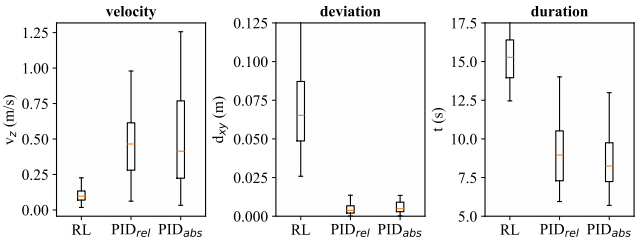

Figure 8: Validation box plots

Figure 8 shows the comparison with PID-based controllers that were designed for position and velocity track-

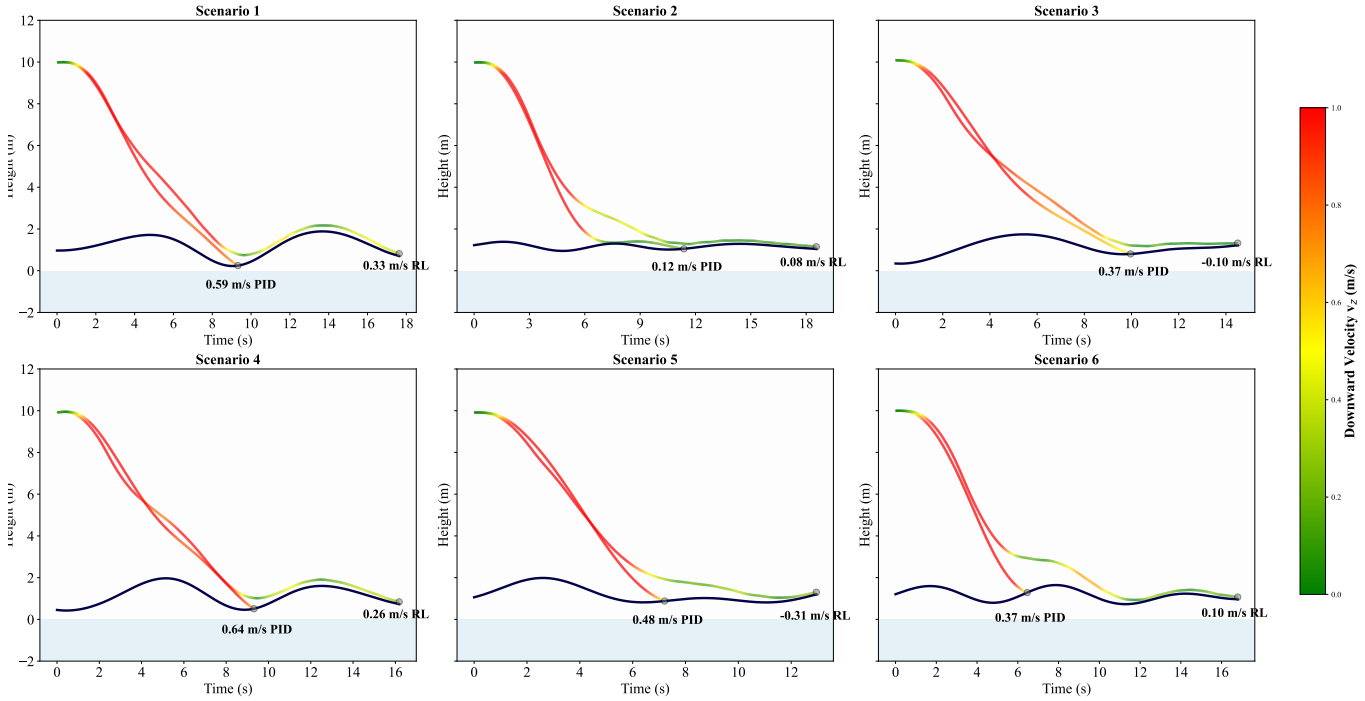

Figure 9: Validation trajectories

ing, namely, PID$_{rel}$ and PID$_{abs}$. The former takes the relative velocity of the drone with respect to the platform, while the latter generates velocity commands based on the position tracking only.

The left plot in Figure 8 compares the vertical descent velocities (v_{d_z}) of the RL framework and the two PID controllers. While the figures show that PID$_{rel}$ works slightly better than the PID$_{abs}$ in terms of downward velocity, the RL framework achieves the lowest median velocity among them, with values close to 0.1 m/s. As the limit for damage sits at around 0.5 m/s, these results indicate that RL shows a superior performance with much smoother landings, even leaving some margin in velocity. The statistics for the reinforcement learning and PID controllers are given in Table 2.

Finally, Figure 9 visualizes the trajectories of a drone controlled by both PID controllers (PID$_{rel}$) and the reinforcement learning framework using real ship data not seen during training across six different scenarios. The y-axis indicates the height (in meters) and the x-axis shows the duration of the flight (in seconds). The trajectories are color-coded to indicate the downward velocity (v_{d_z}) of the drone, with green representing low velocities and red representing high velocities. The impact velocities are indicated for both controllers.

The trajectories controlled by the PID consistently result in landings with high touchdown velocities, despite having relative velocity control. In each scenario, PID tries to slow down when approaching the ship, but does not always succeed in adapting to the ship's motion due to its feedback control structure and UAV inertial. In contrast, the reinforcement learning framework displays significantly different landings. It closely tracks the wave patterns of the ship and appears to wait for good conditions before landing. It consistently shows lower touchdown velocities with stable and controlled flights as opposed to the PID control that often results in impact velocities over 0.5m/s instead of successful landings.

Table 2: Summary Statistics for RL and PID Controllers

Label	Min	Max	Mean	Std Dev
RL-vz	0.0176	0.2265	0.1017	0.0453
PID$_{rel}$-vz	0.0617	1.3303	0.4779	0.2459
PID$_{abs}$-vz	0.0325	1.2563	0.4828	0.3082
RL-xy	0.0258	0.1320	0.0692	0.0243
PID$_{rel}$-xy	0.0001	0.0433	0.0073	0.0091
PID$_{abs}$-xy	0.0002	0.0453	0.0084	0.0097
RL-ttc	12.4600	19.7600	15.3107	1.6582
PID$_{rel}$-ttc	5.9500	14.0100	9.0416	1.9637
PID$_{abs}$-ttc	5.700	12.990	8.5610	1.7039

8 Conclusion

This study demonstrated the effectiveness of a reinforcement learning-based framework for the autonomous landing of the Variable Skew Quad Plane on a moving ship. Several simulations performed with randomized sinusoidal signals revealed that the reinforcement learning could adapt to a wide variety of sinusoidal ship motions and generate realistic acceleration commands that effectively adjust the drone's speed

and trajectory to achieve successful landings.

The validation with real ship data, which incorporated non-sinusoidal and unpredictable movements, confirmed the robustness and adaptability of our framework. Compared to the benchmark controller, reinforcement learning achieved significantly lower impact velocities that resulted in controlled, safer landings, closely tracking the wave patterns of the ship and waiting for an acceptable landing moment.

This work highlighted the capabilities of reinforcement learning for a challenging autonomous landing task, which requires enhanced performance against environmental variability and operational uncertainty. The integration of reinforcement learning as a guidance model within the control loop promotes safer and more reliable autonomous operations.

Future work may expand upon this study by performing real-life experiments and especially investigating the robustness of the proposed algorithm in various sea states. In addition to that, different network structures (e.g., Recurrent Neural Networks) could enhance the results by better learning the patterns in large amounts of sea state data. Lastly, a more detailed analysis of the reward elements may be performed to incorporate ship characteristics further into the objective.

REFERENCES

[1] JeongWoon Kim, Yeondeuk Jung, Dasol Lee, and David Hyunchul Shim. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252. IEEE, 2014.

[2] Kun Li, Peidong Liu, Tao Pang, Zhaolin Yang, and Ben M Chen. Development of an unmanned aerial vehicle for rooftop landing and surveillance. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 832–838. IEEE, 2015.

[3] Yi Feng, Cong Zhang, Stanley Baek, Samir Rawashdeh, and Alireza Mohammadi. Autonomous landing of a uav on a moving platform using model predictive control. *Drones*, 2(4):34, 2018.

[4] Tomaso Maria Luigi De Ponti. Incremental nonlinear dynamic inversion controller for a variable skew quad plane. Master's thesis, Delft University of Technology, 2022.

[5] Oualid Araar, Nabil Aouf, and Ivan Vitanov. Vision based autonomous landing of multirotor uav on moving platform. *Journal of Intelligent & Robotic Systems*, 85:369–384, 2017.

[6] Liguo Tan, Juncheng Wu, Xiaoyan Yang, and Senmin Song. Research on optimal landing trajectory planning method between an uav and a moving vessel. *Applied Sciences*, 9(18):3708, 2019.

[7] Pablo R Palafox, Mario Garzón, João Valente, Juan Jesús Roldán, and Antonio Barrientos. Robust visual-aided autonomous takeoff, tracking, and landing of a small uav on a moving landing platform for life-long operation. *Applied Sciences*, 9(13):2661, 2019.

[8] Ajmal Hinas, Jonathan M Roberts, and Felipe Gonzalez. Vision-based target finding and inspection of a ground target using a multirotor uav system. *Sensors*, 17(12):2929, 2017.

[9] Marcin Skoczylas. Vision analysis system for autonomous landing of micro drone. *acta mechanica et automatica*, 8(4):199–203, 2014.

[10] Bochan Lee, Vishnu Saj, Dileep Kalathil, and Moble Benedict. Intelligent vision-based autonomous ship landing of vtol uavs. *Journal of the American Helicopter Society*, 68(2):113–126, 2023.

[11] Xuancen Liu, Shifeng Zhang, Jiayi Tian, and Longbin Liu. An onboard vision-based system for autonomous landing of a low-cost quadrotor on a novel landing pad. *Sensors*, 19(21):4703, 2019.

[12] Lizhen Wu, Chang Wang, Pengpeng Zhang, and Changyun Wei. Deep reinforcement learning with corrective feedback for autonomous uav landing on a mobile platform. *Drones*, 6(9):238, 2022.

[13] Lirong Zhou, Anton Pljonkin, and Pradeep Kumar Singh. Modeling and pid control of quadrotor uav based on machine learning. *Journal of Intelligent Systems*, 31(1):1112–1122, 2022.

[14] Botao Hu, Lu Lu, and Sandipan Mishra. Fast, safe and precise landing of a quadrotor on an oscillating platform. In *2015 American Control Conference (ACC)*, pages 3836–3841. IEEE, 2015.

[15] JeongWoon Kim, YeonDeuk Jung, DaSol Lee, and David Hyunchul Shim. Landing control on a mobile platform for multi-copters using an omnidirectional image sensor. *Journal of Intelligent & Robotic Systems*, 84:529–541, 2016.

[16] Shyh-Pyng Shue and Ramesh K. Agarwal. Design of automatic landing systems using mixed h2/h1 control. *Journal of Guidance, Control, and Dynamics*, 22(1), 1999.

[17] Yi Feng, Cong Zhang, Stanley Baek, Samir Rawashdeh, and Alireza Mohammadi. Autonomous landing of a uav on a moving platform using model predictive control. *Drones*, 2(4), 2018.

[18] Alireza Mohammadi, Yi Feng, Cong Zhang, Samir Rawashdeh, and Stanley Baek. Vision-based autonomous landing using an mpc-controlled micro uav on a moving platform. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 771–780, 2020.

[19] Man Yuan, Chang Wang, Pengpeng Zhang, and Changyun Wei. *PID with Deep Reinforcement Learning and Heuristic Rules for Autonomous UAV Landing*, pages 1876–1884. Springer Singapore, Singapore, 03 2023.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[21] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[22] Riccardo Polvara, Massimiliano Patacchiola, Sanjay Sharma, Jian Wan, Andrew Manning, Robert Sutton, and Angelo Cangelosi. Toward end-to-end control for uav autonomous landing via deep reinforcement learning. In *2018 International conference on unmanned aircraft systems (ICUAS)*, pages 115–123. IEEE, 2018.

[23] Seongheon Lee, Taemin Shim, Sungjoong Kim, Junwoo Park, Kyungwoo Hong, and Hyochoong Bang. Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 108–114. IEEE, 2018.

[24] Alejandro Rodriguez-Ramos, Carlos Sampedro, Hriday Bavle, Paloma de la Puente, and Pascual Campoy. A deep reinforcement learning strategy for uav autonomous landing on a moving platform. *Journal of Intelligent & Robotic Systems*, 93:351–366, 2019.

[25] Jingyi Xie, Xiaodong Peng, Haijiao Wang, Wenlong Niu, and Xiao Zheng. Uav autonomous tracking and landing based on deep reinforcement learning strategy. *Sensors*, 20:5630, 2020.

[26] Vishnu Saj, Bochan Lee, Dileep Kalathil, and Moble Benedict. Robust reinforcement learning algorithm for vision-based ship landing of uavs. *arXiv preprint arXiv:2209.08381*, 2022.

[27] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017.

[28] TML De Ponti, EJJ Smeur, and BWD Remes. Incremental nonlinear dynamic inversion controller for a variable skew quad plane. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 241–248. IEEE, 2023.

[29] HJ Karssies and C De Wagter. Extended incremental non-linear control allocation (xinca) for quadplanes. *International Journal of Micro Air Vehicles*, 14:17568293211070825, 2022.

[30] Robin Ferede, Guido de Croon, Christophe De Wagter, and Dario Izzo. End-to-end neural network based optimal quadcopter control. *Robotics and Autonomous Systems*, 172:104588, 2024.

[31] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[32] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[34] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.