drones

MDPI

*Article*

# A Robust Strategy for UAV Autonomous Landing on a Moving Platform under Partial Observability

**Godwyll Aikins, Sagar Jagtap and Kim-Doang Nguyen ***

Department of Mechanical and Civil Engineering, Florida Institute of Technology, Melbourne, FL 32901, USA; gaikins2015@my.fit.edu (G.A.)
* Correspondence: knguyen@fit.edu

**Abstract:** Landing a multi-rotor uncrewed aerial vehicle (UAV) on a moving target in the presence of partial observability, due to factors such as sensor failure or noise, represents an outstanding challenge that requires integrative techniques in robotics and machine learning. In this paper, we propose embedding a long short-term memory (LSTM) network into a variation of proximal policy optimization (PPO) architecture, termed robust policy optimization (RPO), to address this issue. The proposed algorithm is a deep reinforcement learning approach that utilizes recurrent neural networks (RNNs) as a memory component. Leveraging the end-to-end learning capability of deep reinforcement learning, the RPO-LSTM algorithm learns the optimal control policy without the need for feature engineering. Through a series of simulation-based studies, we demonstrate the superior effectiveness and practicality of our approach compared to the state-of-the-art proximal policy optimization (PPO) and the classical control method Lee-EKF, particularly in scenarios with partial observability. The empirical results reveal that RPO-LSTM significantly outperforms competing reinforcement learning algorithms, achieving up to 74% more successful landings than Lee-EKF and 50% more than PPO in flicker scenarios, maintaining robust performance in noisy environments and in the most challenging conditions that combine flicker and noise. These findings underscore the potential of RPO-LSTM in solving the problem of UAV landing on moving targets amid various degrees of sensor impairment and environmental interference.

**Keywords:** deep reinforcement learning; unmanned aerial vehicles; partial observability; model-free control

check for updates

**Citation:** Aikins, G.; Jagtap, S.; Nguyen, K.-D. A Robust Strategy for UAV Autonomous Landing on a Moving Platform under Partial Observability. *Drones* **2024**, *8*, 232. https://doi.org/10.3390/drones8060232

Academic Editor: Bo Li

Received: 2 May 2024
Revised: 21 May 2024
Accepted: 27 May 2024
Published: 30 May 2024

**Copyright:** © 2024 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

Over the past decade, uncrewed aerial vehicles (UAVs) and their applications have gained increased importance and attention in various domains, such as military operations, search and rescue missions, agriculture, and surveillance [1–3]. However, one major challenge in UAV operations is landing on a dynamic target, such as a moving vehicle or a ship. The capability to land on a moving target has numerous practical applications, including delivering supplies to moving vehicles [4], providing aerial support to moving troops, and inspecting mobile infrastructure [5]. In [6–8], landing on a moving platform was tackled with several techniques that solved the problem analytically and performed well under restricted conditions and assumptions.

The robustness of autonomous UAV landings on mobile platforms hinges on the accurate localization of both the UAV and the target, adherence to a predetermined flight trajectory despite environmental disturbances like wind, and the development of a system that autonomously executes landings without external aids, such as motion capture systems or fiducial markers.

Achieving full autonomy and robustness typically requires UAV systems equipped with visual and inertial sensors. The integration of the sensors enables onboard state estimation and detection of the landing platform. Although visual-inertial odometry (VIO)

and visual servoing have been extensively researched for UAV landings on moving targets, a gap remains in addressing the challenges posed by scenarios with limited observability, such as missing detection of the target due to occlusion, weather, or motion blur, which pose serious threats to the methods. In [9], researchers formulated a visual servoing technique for the autonomous landing of UAVs on a platform capable of speeds up to 15 km/h. Of the 19 conducted trials, 17 were successful. The two unsuccessful attempts were attributed to the platform not being detected. Overcoming issues such as intermittent platform detection is essential for ensuring the safe and reliable landing of drones on mobile targets.

Equally important is addressing sensor failure. Inertial measurement units (IMUs), GPS, compasses, barometers, and other sensors that enable autonomous UAV flight are susceptible to a number of malfunctions, including magnetic field interference, turbulence-induced disconnections, or sensor malfunctions that lead to altered or missing sensor data. The literature suggests fault-tolerant methods that involve distinct modules for rapid sensor failure detection and subsequent system reconfiguration to either compensate for lost data or switch to backup systems [10]. Nevertheless, a paucity of literature explores the occurrence of sensor failures amid the execution of landing maneuvers.

In this paper, we emphasize the significance of addressing partially observable scenarios in landing drones on moving targets and propose fusing robust policy optimization (RPO) with long short-term memory (LSTM) as an appropriate framework. Adopting an end-to-end fault-tolerant control strategy enables the UAV to address sensor failures or detection lapses by processing all fault-tolerant control stages through a unified model. Additionally, employing a learning-based control approach facilitates the training of neural networks to address a variety of fault scenarios. While existing research explores the cooperative interaction between UAVs and uncrewed ground vehicles (UGVs), our approach assumes no communication or coordination occurs during landing.

The contributions of this work with respect to prior literature are as follows: (a) We formulate an end-to-end RPO-LSTM reinforcement learning framework that tackles the challenging task of landing a UAV with missing sensor data on a moving target. (b) We validate the proposed approach through extensive simulation experiments, which include partial observability due to factors such as sensor failure or sensor noise. (c) We illustrate that adding memory to a UAV RL agent leads to higher rewards compared to conventional deep reinforcement learning algorithms, thus substantially improving system performance.

*Structure*

The remainder of the paper is organized as follows. Section 2 discussed previous works on autonomous UAV landing. Section 3 delves into the technical framework of our method and details the formulation of the UAV landing task as a POMDP and the network architecture and training. Section 4 outlines the use of the NVIDIA Isaac Gym simulator, describes the training process, and establishes performance metrics. The results are presented to demonstrate RPO-LSTM's performance compared to other methods under diverse partial observability scenarios. Section 5 dissects the impact of individual components in the RPO-LSTM architecture. Finally, Section 6 provides a conclusion to the paper, summarizing the findings and suggesting avenues for future investigation.

## 2. Literature Review

The challenge of landing UAVs on moving targets is an active area of research, with various approaches developed to tackle the complexities of dynamic environments and limited sensor data. Researchers seek to enable robust, fully autonomous UAV landings on diverse moving platforms. Given the multifaceted nature of the challenge, much of the literature focuses on specialized solutions for individual components. Proposed solutions aim to achieve the precise localization of both the UAV and the target, ensure robust trajectory following, or develop systems capable of autonomous landing without the need for external infrastructure.

Extensive research has been dedicated to exploring various methods for the state estimation of the UAV and UGV. Prior work utilized motion capture systems [11], GPS [12], or computer vision [13] to obtain pose estimates of the UAV and UGV. Recent advancements in computer vision have enabled onboard localization and pose estimation, reducing the dependency on external infrastructure for maneuver execution. Early research by [13–15] relied on visual fiducial systems to facilitate relative pose estimation between the drone and the target, simplifying the maneuver to just requiring a camera and a marker externally. Present-day research has pivoted toward machine learning techniques for platform detection and relative pose estimation, leading to a proliferation of visual servoing methods [9].

The most common challenge with vision-based systems in UAV landing on a moving target is partial observability, particularly when detecting highly dynamic moving targets. Visual servoing methods, which rely on visual feedback for control, are particularly affected by partial observabilities, as they require the landing platform to be visible throughout the entire landing maneuver. Partial observability poses significant difficulties in accurately tracking the moving platform and maintaining stable and safe landing trajectories.

A diverse array of control strategies has been employed to govern UAVs during landing maneuvers. The strategies range from classical methods, such as proportional–integral–derivative (PID) [13] and linear quadratic regulator (LQR) [16], to contemporary approaches, like model predictive control (MPC) [17], and even learning-based techniques, such as reinforcement learning [18]. The methodologies are instrumental in executing robust trajectory tracking amidst uncertainties and disturbances. Nevertheless, the majority of literature that utilizes classical control methods often overlooks notable disturbances. To enhance the resilience of classical methods against disturbances and uncertainties, numerous adaptive mechanisms are required to be integrated into the control loop. The mechanisms dynamically adjust the controller parameters in response to the observed behavior of the system. In pursuit of mitigating external disturbances, some studies have turned to MPC. Notably, the work in [17] incorporated a boundary layer sliding controller, enabling a quadcopter to land under wind conditions with speeds reaching up to 8 m/s.

Recent studies have increasingly focused on learning-based methods, especially reinforcement learning (RL), for the task of landing on moving platforms. The shift aims to address the inherent complexities associated with model-based approaches. RL-based control systems are noted for their robustness, adapting to a wide range of changes in system dynamics and unexpected disturbances, areas where static model-based controllers falter. Prior works have developed RL algorithms that are resilient to perturbations in the environment dynamics [19–21], observations [22–25], and actions [26]. The robust RL algorithms can learn optimal policies that maximize returns in the worst-case environments. For instance, ref. [27] introduced a resilient RL algorithm capable of executing end-to-end, vision-based control for landing on a moving ship platform, even under windy conditions. Similarly, ref. [18] proposed a robust RL strategy designed to accommodate variations in system dynamics while landing on a moving target. While RL is predominantly applied for robust, end-to-end control in landing scenarios, research exploring its use in managing partial observability caused by missing sensor data remains scant.

The literature addressing the issue of missing observations is sparse. Some studies, such as [7,12], have addressed the challenge of temporarily missing detections by employing an extended Kalman filter (EKF). The methods predict the pose of a moving target using a dynamic model that assumes constant velocity. However, the effectiveness of the approach is considerably constrained by its assumptions. Recent studies, along with the uncertainties they address, are detailed in Table 1.

To fill the gap between optimality and control with partial observabilities, this work proposes to integrate a type of RNN, LSTM [28], into the RPO [29] architecture to capture long-range dependencies in sequential sensing data and handle variable-length signal sequences. The main goal is to discover missing patterns in sensing signals that have missing readings due to faulty sensors and to achieve the desired landing performance despite the partial observabilities in the environment. To highlight the significance of

the memory element in addressing the issue, our research contrasts traditional methods employing an EKF with our approach and evaluates it against the baseline proximal policy optimization (PPO) algorithm as referenced in [30]. PPO is a simple RL algorithm that is easy to implement and tune while achieving state-of-the-art performance, stability, reliability, and sample efficiency. When using RL to learn the optimal control policy with UAVs, PPO has become the preferred option. For instance, researchers deployed an RL PPO-based algorithm that enables a UAV to navigate through unknown/random environments [31]. PPO was also used as an uncrewed traffic manager to lead autonomous uncrewed aircraft systems to their destinations while avoiding obstacles through continuous control. In [32], a fully autonomous UAV controller was developed to steer a drone toward a moving target using PPO.

**Table 1.** Comparison of control algorithms in related works and their respective uncertainties and disturbances investigated.

| Paper | Control Algorithm | Uncertainty/Disturbance Tackled |
|---|---|---|
| [17] | MPC | Wind |
| [27] | DDPG-RL | Wind |
| [33] | DrQv2-RL | Noisy Sensor |
| [7] | PID | Missing Platform Detection |
| [13] | PID | Wind |
| [12] | PID | Noisy Sensor |
| [34] | MPC | Wind |
| [15] | Visual Servo | Wind |

## 3. Methodology

The methodology section, which outlines the technical approach to address the UAV landing problem under partial observability, starts with Section 3.1, providing a basis for MDPs and POMDPs. Section 3.2 defines the task as a POMDP, specifying the observations, actions, and reward structure. Section 3.3 presents the innovative actor–critic design with a unique information flow separation, along with LSTM integration for sequential data. Section 3.4 details the RPO-based learning process, including policy updates, exploration, and LSTM training. Finally, Section 3.5 highlights the advantages of the approach and provides an algorithm outlining the RPO-LSTM training process.

### *3.1. Preliminaries*

The Markov decision process (MDP) is a mathematical framework for a stochastic environment in which the state of the system is observable and follows the Markov property. The MDP is defined by a 4-tuple $(S, A, P, R)$: state space ($S$), action space ($A$), transiton probability ($P$), and reward ($R$). However, a partially observable Markov decision process (POMDP) describes situations in which the system's state is not entirely observable. POMDP is defined by a 6-tuple $(S, A, P, R, O, \Omega)$, where $S$, $A$, $P$, and $R$ are identical to the MDP but with an additional observation space ($O$) and observation model ($\Omega$). In a POMDP, the agent cannot directly observe the underlying state. An illustration of the MDP and POMDP as dynamic decision networks are shown in Figure 1. One way of dealing with POMDPs is by giving memory to the agent. Recurrent reinforcement learning (RRL) is a special kind of RL that integrates RNNs into the RL framework. RNNs allow an RL agent to keep and update memory of past states and actions, allowing for a better understanding of the current state and making more informed decisions. RRL is especially useful in sequential decision-making tasks, in which the recent action depends on the previous state and action.

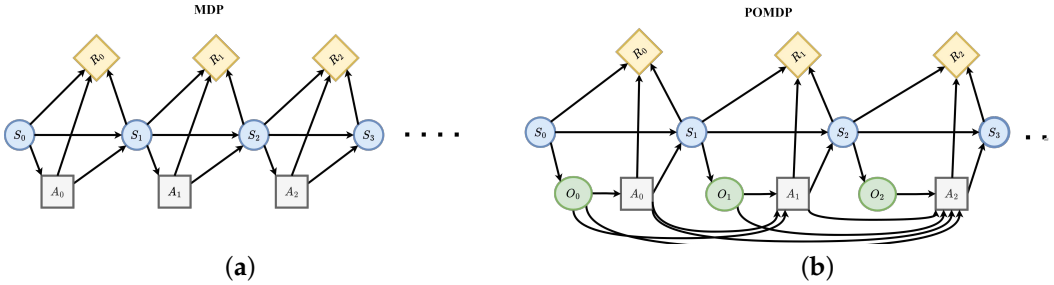


**Figure 1.** Illustration of (**a**) MDP dynamic decision networks and (**b**) POMDP as dynamic decision networks.

### *3.2. Problem Formulation*

The objective of this paper is to formulate an innovative RL architecture for UAVs to land on a dynamic target with partial observability. The new RL architecture is grounded in the idea of integrating agent memory, enabled by LSTM, into the RPO framework. The work also evaluates the new integrative RPO-LSTM architecture as a desirable controller for a multi-rotor UAV in the presence of disturbances and uncertainties, which entails developing a control policy that maps the current state of the UAV and mobile platform to the UAV's thrust commands to regulate its altitude and orientation for successful landing. We formulate the control problem as a POMDP where the agent must make decisions based on improper observations rather than the true state of the environment. In the following sections, observations, actions, and rewards are explained in further detail in the context of the task.

#### 3.2.1. Observations

During each time step, the RL agent is supplied with a sequence of observations that is composed of the drone's current state and position relative to the moving platform. The observations are contained in a $13 \times 1$ vector $[d,\ v,\ \omega,\ q]^\top$, where $d$, $v$, $\omega$, and $q$ represent the drone's 3D distance to the moving platform, 3D linear velocity, 3D angular velocity, and 4D quaternion, respectively.

#### 3.2.2. Actions

Our UAV model has four control inputs: two clockwise and two counterclockwise rotors. The actor network takes in the observation vector and produces a $4 \times 1$ action vector that comprises the thrust values assigned to each rotor, representing the control input. The action space is denoted as $a_t = [T_1, T_2, T_3, T_4]$, where $a_t$ is the vector at time step $t$ and $T_i$ represents the thrust magnitude for each rotor. The control commands vary individually for each rotor, which enables flexible and dynamic flight behaviors.

The actor network incorporates a Tanh activation function at the concluding layer to confine the range of control commands, ensuring compatibility with the physical capabilities of the UAV. Subsequently, the agent executes the derived actions in a continuous cycle until the defined goal is successfully attained.

#### 3.2.3. Rewards

The reward function considers various aspects of the landing maneuver. The distance between the drone and the moving platform is included in the reward function. Moreover, a shaping function is employed to distinguish the significance of minimizing the position and velocity with respect to the moving platform and the generated actions. The agent learns to minimize the position relative to the moving platform and subsequently optimize its actions to produce smoother velocity references, which leads to a less aggressive movement. The reward components are described as follows.

- The distance reward ($r_{dist}$) represents the distance between the UAV and the dynamic platform, encouraging the UAV to maintain appropriate proximity:

$$r_{\text{dist}} = \frac{1}{1 + |\text{distance}|^2} \tag{1}$$

- The yaw reward ($r_\Gamma$) aims to discourage excessive yawing and promote stable heading orientation when close to the platform:

$$r_\Gamma = \frac{1}{1 + |\text{yaw}|^2} \tag{2}$$

- The pitch reward ($r_\phi$) is associated with the UAV's pitch when near the moving platform, emphasizing the need to keep the UAV upright during close interaction:

$$r_\phi = \frac{1}{1 + |\text{pitch}|^2} \tag{3}$$

By considering the reward components, the UAV's behavior is effectively guided and optimized for improved performance and safety. The reward function also considers the quadcopter's orientation during the landing process. The overall reward function is summarized by the following expression:

$$R_t = r_{\text{dist}} + r_{dist}(1 + r_\Gamma + r_\phi) \tag{4}$$

This multiplicative reward structure underscores the significance of the UAV's orientation in correlation with its distance from the target. The underlying principle is that as the UAV nears the landing platform, maintaining the correct orientation becomes increasingly imperative. Conversely, when the UAV is further away, precise orientation is less crucial.

### *3.3. Network Architecture*

To achieve the goal of successfully landing a UAV onto a moving platform under conditions of limited observability, we establish an actor–critic RL architecture. The architecture is composed of two distinct components: an actor network and a critic network. The actor, characterized by parameters $\theta$, is designed to take inputs $[d,\ v,\ \omega,\ q]^\top$, which represent the UAV's 3D distance to the moving platform, 3D linear velocity, 3D angular velocity, and the 4D quaternion, respectively. The actor then generates actions containing the thrust values of the four individual rotors, as outlined in Section 3.2.2, guided by a policy denoted as $\pi_\theta$. On the other hand, the critic serves as an action-value evaluator, which supplies a value that gauges the effectiveness of the computed action within the given state. During the training process, the two networks are collectively optimized.

The key innovation of our architecture, as compared to the traditional actor–critic framework, is the separation of observation and state in accordance with the actor and the critic. Specifically, we feed the observations to the actor and the states to the critic during the training process. The states encapsulate accurate ground-truth information about the environment, including the UAV's state and the states of all sensors. Conversely, the observations comprise sensor data perceived by the robot, which, in the POMDP setting, contains noise, inaccuracies, and missing information. The innovative approach of feeding the actor with the POMDP observations and the critic with states serves a distinct purpose that empowers the actor to deduce the actual states of the UAV and moving platform based on the partial observations received. The complete network architecture is shown in Figure 2.

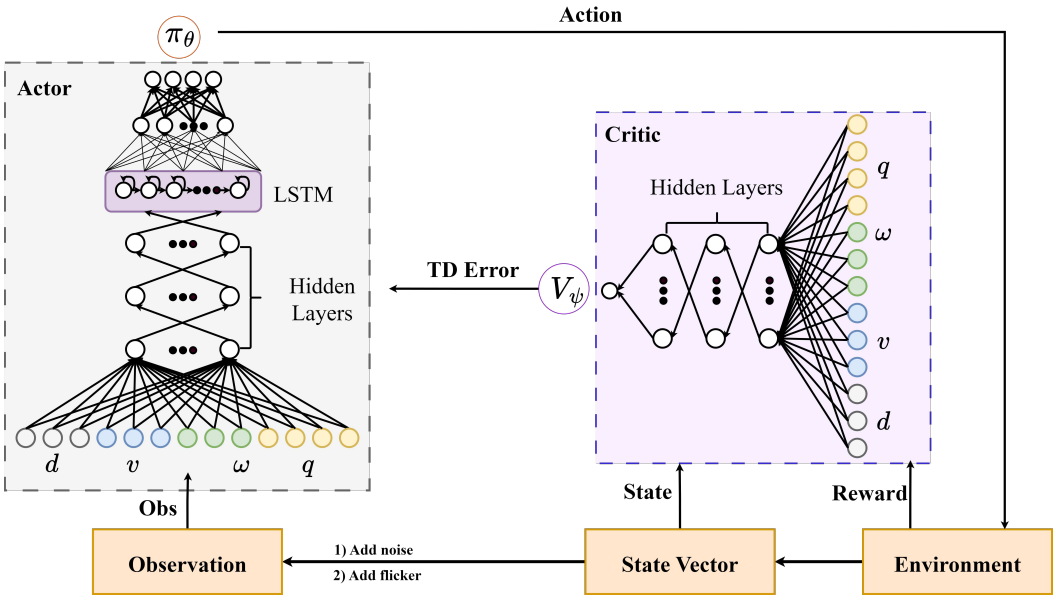


**Figure 2.** The architecture of RPO-LSTM, highlighting the distinctive separation of information flow between the actor and critic networks. The critic network benefits from additional observations, contributing to robust value estimation, while the actor network utilizes partial observations, enabling adaptive and efficient decision-making in dynamic environments.

#### 3.3.1. Actor

The observations taken by the actor are processed through a series of hidden layers with substantial depth. The outputs from these deep neural networks are then fed into an LSTM network, which is responsible for extracting temporal patterns within the observation and learning the policy for the precision landing task. The LSTM network maintains a memory cell that can store information across time steps, allowing the network to learn long-term dependencies and make decisions based on previous observations. The output of the LSTM network is in the form of a probability distribution over possible actions. The agent selects an optimal action to take for the UAV. The selected action is then executed in the environment, which then responds by providing a new set of observations.

#### 3.3.2. Critic

After receiving the state from the environment, as described in Section 3.2.1, it undergoes processing through a multi-layer perceptron (MLP). The MLP's role is to approximate the value function associated with the states. Subsequently, the actor updates its policy based on the output of the value function provided by the critic. The value function approximated by the critic is defined as

$$\begin{aligned} V_\psi(s) &= \hat{\mathbb{E}}_t[\mathcal{G}_t \mid S_t = s] \\ &= \hat{\mathbb{E}}_t\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\ &= \hat{\mathbb{E}}_t[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) \mid S_t = s] \\ &= \hat{\mathbb{E}}_t[R_{t+1} + \gamma \mathcal{G}_{t+1} \mid S_t = s] \\ &= \hat{\mathbb{E}}_t\left[R_{t+1} + \gamma V_\psi(s_{t+1}) \mid S_t = s\right] \end{aligned} \tag{5}$$

In this equation, $\mathcal{G}_t$ represents the cumulative sum of rewards starting from time $t$ onward, $R_t$ is the immediate reward at time step $t$, and $\gamma$ is the discount factor, which determines the weight given to future rewards. The equation establishes a recursive relationship expressing the value function ($V_\psi(s)$) as a measure of the desirability of a given state with respect to expected future rewards.

### *3.4. Network Training*

The proposed UAV landing algorithm is an on-policy method that seeks to improve the training stability of a control policy by limiting the change made to the policy at each training epoch by avoiding having too large of a policy update. In our training of the deep reinforcement learning agent, we leverage the following policy-gradient objective function:

$$L^{\mathrm{PG}}(\theta) = \hat{\mathbb{E}}_t[\log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{6}$$

where $\hat{\mathbb{E}}_t$ denotes the empirical return over a batch of samples, $\pi_\theta$ is a stochastic policy, and $\hat{A}_t$ is the estimator of the advantage function at time $t$. By taking the gradient ascent of the equation above, the agent takes actions that lead to higher rewards and avoids harmful actions.

The objective function of the gradient method is optimized by constraining the policy updates to ensure that the new policy is not too different from the old policy. The idea is to constrain the policy change in a small range using a clip. This new objective function, the clipped surrogate objection function, is defined as follows:

$$L^{\mathrm{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{7}$$

where $r_t(\theta)$ is the probability ratio of the policy at time $t$ with parameter $\theta$ and the policy at time $t$ and $\epsilon$ is a clipping parameter that controls the size of the update. The probability ratio function is designed as follows:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \tag{8}$$

where $\pi_\theta(a_t|s_t)$ is the probability of taking action $a_t$ in state $s_t$ under policy $\pi_\theta$ with parameter $\theta$ and $\pi_{\theta_{old}}(a_t|s_t)$ is the probability of taking the same action under the old policy with parameter $\theta_{old}$. The ratio represents the change in the probability of taking an action under the new policy compared to the old policy.

The first term of (7), $r_t(\theta)\hat{A}_t$, replaces the logarithmic probability typically used in (6). The ratio $r_t(\theta)$ is multiplied by the advantage $\hat{A}_t$ to obtain the left part of the new objective function. However, without a constraint, a large change in the probability of taking an action under the new policy compared to the old policy can lead to a significant policy gradient step and an excessive policy update. Therefore, this algorithm imposes a constraint that limits the size of policy updates by clipping the probability ratio to a value between $1-\epsilon$ and $1+\epsilon$. This results in a clipped surrogate objective function that balances between the benefits of a large update and the risks of destabilizing the policy. The minimum between the clipped and unclipped objective is taken to make the final objective a lower bound of the unclipped objective.

The critic parameters are updated using

$$\psi \leftarrow \psi + \alpha_v \cdot \frac{1}{n}\sum_i \nabla_\psi V_\psi(s_t)(y_t - V_\psi(s_t)) \tag{9}$$

Here, $\psi$ represents the value function's parameter vector, $\alpha_v$ is the learning rate, $n$ is the number of collected samples, $\nabla_\psi V_\psi(s_t)$ is the gradient of the value function, and $y_t$ is the target value at time step $t$.

The actor parameters are updated using

$$\theta \leftarrow \theta + \alpha_\pi \cdot \frac{1}{n}\sum_k \nabla_\theta \min(w_t(\theta)A_t, \mathrm{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \tag{10}$$

Here, $\theta$ represents the policy's parameter vector, $\alpha_\pi$ is the learning rate, $n$ is the number of collected samples, $\nabla_\theta$ is the gradient of the policy, $w_t(\theta)$ measures the policy difference $w_t(\theta) = \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$, and $A_t$ is the advantage function.

### *3.5. Properties of the Methodology*

Traditional approaches to UAV landings on mobile platforms typically utilize a modular framework comprising distinct components for UAV state estimation, moving target estimation, trajectory planning, and UAV control, as exemplified in Figure 3. In contrast, our method proposes an end-to-end control policy that consolidates the modules into a singular neural network. Our work not only encompasses these tasks but also extends to managing partial observabilities adeptly. To achieve end-to-end control, we synthesize existing methods into an integrated framework tailored for landing on dynamic targets under conditions of partial observability. We draw upon [29] to enhance exploration strategies, employ LSTM networks [28] for capturing temporal dependencies in sequential sensor data, and utilize asymmetric actor-critic methods akin to [35] to enable the network to deduce missing information. The contribution of the methodologies to our framework is explained in subsequent sections. Algorithm 1 demonstrates how these modules are integrated and trained in unison.

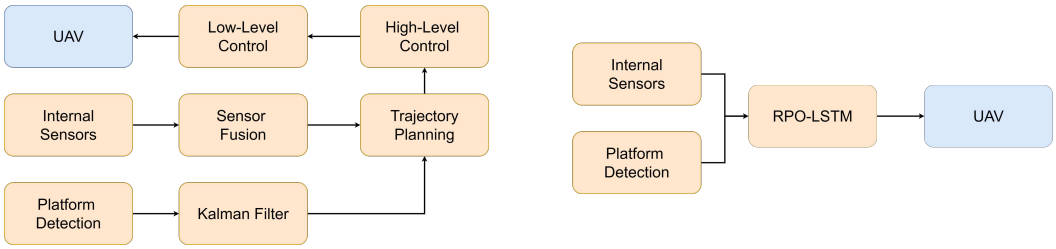


**Figure 3.** A schematic representing a typical framework using classical control methods vs. our end-to-end RL framework.

#### 3.5.1. Fostering Enhanced Exploration Strategies

The inherent stochasticity in policy gradient methods gradually diminishes during training. This leads to a reduction in the exploratory nature of the policy. To stimulate a more exploratory approach, we implemented a distribution mechanism for representing continuous actions, complementing the foundational parameterized Gaussian distribution. In our work, the policy network's output still encompasses the Gaussian distribution's mean and standard deviation. A key innovation is the integration of a random value $z \sim U(-\alpha, \alpha)$ into the mean $\mu$, resulting in a perturbed mean $\mu' = \mu + z$. Subsequently, actions are sampled from the perturbed Gaussian distribution $a \sim \mathcal{N}(\mu', \sigma)$. Notably, the resultant distribution displays a broader shape compared to the conventional Gaussian. As a consequence, the sampled values exhibit a wider dispersion around the mean, in contrast to the tighter clustering observed in the case of the standard Gaussian distribution.

#### 3.5.2. Capturing Long-Range Dependencies in Sequential Sensing Data

One of the key contributions of this work is the ability to deal with imperfect or flickering data under partial observability. This goal is accomplished by embedding the LSTM network into the RL actor network. LSTM is a type of RNN that is commonly used in machine-learning research for learning spatial-temporal features and sequence modeling. Unlike traditional RNNs, LSTM cells are designed to selectively retain or forget information from previous time steps as shown in Figure 4. This allows our RL agent to effectively learn from long-term dependencies in sequential sensing data despite flickering due to faulty sensors.

Each LSTM cell is composed of several gates that control the flow of information: forget gates, input gates, and output gates. The forget gate determines which information

from the previous cell state to forget, while the input gate decides which new information to add to the cell state. The output gate controls which information from the current cell state to output to the next time step. Using this gated-like structure, the RL agent can selectively transmit sequential sensing data while preserving temporal information within each cell state.

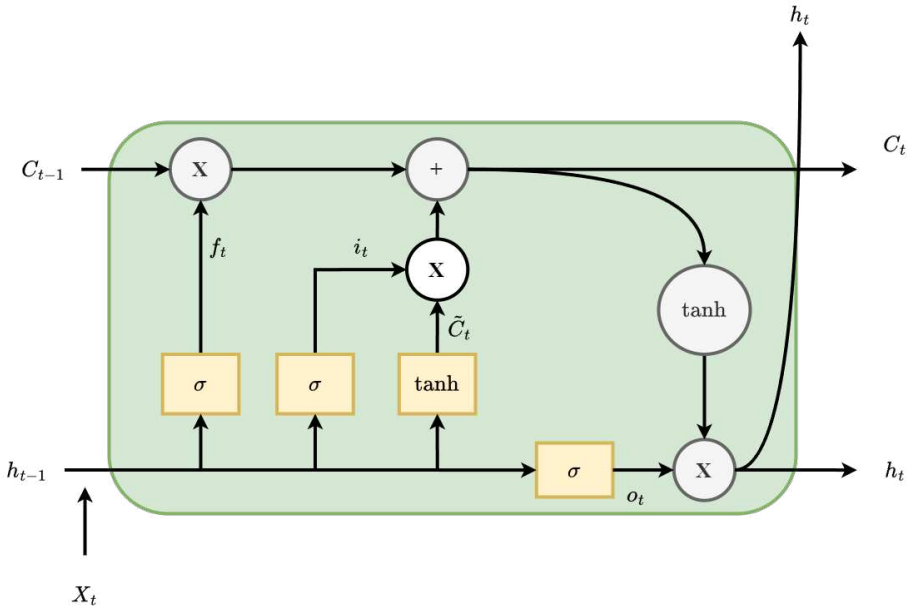


**Figure 4.** The LSTM cell employs specialized gates to control information flow. The 'Forget Gate' $f_t$ determines what to retain or forget from the previous cell state, while the 'Input Gate' $i_t$ and 'Candidate Cell State' $\tilde{C}_t$ determine new information to add. The cell state $C_t$ updates accordingly. The 'Output Gate' $o_t$ regulates the information to be output as the hidden state $h_t$, making LSTM effective for capturing long-term dependencies in sequential data.

#### 3.5.3. Learning to Infer Missing Information

By separating the flow of data into the actor and the critic in our RL framework, we leverage the privilege that the critic can tap into the state information that is not available to the actor, which can only partially observe the measurements. This encompasses noiseless observations and supplementary data into the critic network. The incorporation of these additional inputs serves a dual purpose. Firstly, it streamlines the task of acquiring accurate value estimates, as fewer aspects need to be inferred. Secondly, this streamlined learning process indirectly teaches the actor to adeptly infer the absent information, thereby enhancing its capability to handle partial observations effectively. The key link between the actor and the critic is the advantage function. The advantage of a state–action pair $A(s, a)$ quantifies how much better the chosen action is compared to the average expected value at that state. It is calculated as the difference between the actual value (or Q-value) and the predicted value (V-value) of the critic.

$$A(s, a) = Q(s, a) - V(s) \tag{11}$$

Fundamentally, the actor's decision-making process is influenced by the disparity in advantage values among different actions. When an action demonstrates a substantial advantage, implying its potential for more favorable outcomes, the actor adjusts its decision strategies. This translates to an increased likelihood of selecting the advantageous action when encountering similar circumstances. Consequently, the actor concentrates on actions with higher reward potential, thereby enhancing the intelligence and efficacy of its decision-making. For our variation of RPO, the advantage is solved with a temporal difference (TD)

error estimation, which is the difference between the predicted and actual output of the value function over successive time steps.

$$A(s,a) = R_t(s,a) + \gamma \cdot V_\psi(s') - V_\psi(s) \tag{12}$$

where $R_t(s,a)$ is the immediate reward (4) associated with taking action $a$ in state $s$. The $\gamma$ corresponds to the discount factor. The function $V_\psi(s)$ denotes the value function assigned to state $s$ formulated in (5), while $V_\psi(s')$ signifies the value function assigned to the subsequent state $s'$ resulting from the execution of action $a$ in state $s$.

**Algorithm 1** Training of RPO-LSTM

Initial policy parameters $\theta_0$, initial value function parameters $\psi_0$, LSTM hidden states
**while** not done **do**
  **for** each environment step **do**
    $\mu, \sigma \leftarrow \pi_\theta(\cdot|o_t)$
    $a_t \sim \mathcal{N}(\mu, \sigma)$
    $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$
    $r_t \sim R(s_t, a_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, o_t, s_{t+1})\}$
  **end for**
  **for** each state $s_t$ and observation $o_t$ **do**
    $\mu, \sigma \leftarrow \pi_\theta(\cdot|o_t)$
    $z \sim U(-\alpha, \alpha)$
    $\mu' = \mu + z$
    $prob \leftarrow \mathcal{N}(\mu', \sigma)$
    $logp \leftarrow prob(a_t)$
  **end for**
  $\theta_{\text{old}} \leftarrow \theta$
  **for** each update step **do**
    Reset LSTM hidden states at the start of each episode or sequence
    Sample $n$ samples $\{(s_t, a_t, r_t, o_t, s_{t+1})\}$ from $\mathcal{D}$
    Update value function:
    **for** each $(s_t, a_t, r_t, s'_t)$ **do**
      Compute target values $y_t$ using Temporal Difference ($\lambda$)
    **end for**
    $\psi \leftarrow \psi + \alpha_v \cdot \frac{1}{n} \sum_i \nabla_\psi V_\psi(s_t)(y_t - V_\psi(s_t))$
    Update policy:
    **for** each $(s_t, a_t, r_t, o_t, s_{t+1})$ **do**
      Compute advantage $A_t$ using $V_\psi$ and Generalized Advantage Estimation
    **end for**
    $w_t(\theta) \leftarrow \frac{\pi_\theta(a_t|o_t)}{\pi_{\theta_{\text{old}}}(a_t|o_t)}$
    $\theta \leftarrow \theta + \alpha_\pi \cdot \frac{1}{n} \sum_i \nabla_\theta \min(w_t(\theta)A_t, \text{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon)A_t)$
    Perform Backpropagation Through Time (BPTT) for LSTM update
  **end for**
**end while**

## 4. Simulation Setup and Results

In this section, we provide a comprehensive overview of the simulation environments that were meticulously configured to conduct our simulation experiments. We delve into the specifics of designing the environments to replicate the conditions under which the RPO-LSTM algorithm operates to guarantee a relevant and rigorous testing ground. Additionally, we outline the training regimen of the RPO-LSTM algorithm, detailing the parameters and computational resources utilized, followed by a presentation of the experimental procedures, including the various scenarios and metrics employed to evaluate the performance of the RPO-LSTM algorithm in simulated environments.

### *4.1. Simulation Setup*

To validate the proposed framework, we conducted experiments using the NVIDIA Isaac Gym, which is a GPU-accelerated high-fidelity physics simulator that has demonstrably facilitated successful sim-to-real transfer for various robots, as evidenced by research on drones [36], manipulators [37], legged robots [38], and humanoid robots [39]. In this simulation, we created an environment that includes both the UAV and the moving platform, which is a Clearpath Husky UGV in our case.

To train a robust UAV agent, we intentionally constructed partially observable scenarios by modifying the flow of data to the RL algorithm. Specifically, we created POMDPs consisting of an MDP version and three additional POMDP versions of the task. The MDP version had a fully observable state space, while the POMDP versions simulated different scenarios that affected the observability of the landing maneuver. The first POMDP scenario included sensor noise that gradually increased over time. The second simulated a case in which remote sensor data were lost during long-distance transmission, which we denoted as flicker; the entire $13 \times 1$ observations vector experienced intermittent zeroing out with varying frequency. The third scenario combined the effects of noise and flicker and created a composite challenge that reflected real-world complexities. We leveraged the Isaac Gym's flexibility to manipulate sensor sampling rates to make the controller robust to varying sensor latencies. During training, we randomized the sampling frequency of each simulated sensor within a range of 10 Hz to 60 Hz. The approach exposed the controller to a diverse set of conditions, including those that may arise from real-world sensor delays, ensuring a more comprehensive evaluation of its performance.

### *4.2. Training Details*

NVIDIA Isaac Gym's parallel training capabilities make training multiple agents simultaneously using GPU acceleration possible. Using such capabilities, we trained 4098 agents in parallel, which reduced the training time to 15 min. Figure 5 illustrates the parallel training framework in the environment. The agents were trained for a total of 500 iterations and experienced about 33 million steps in 15 min.

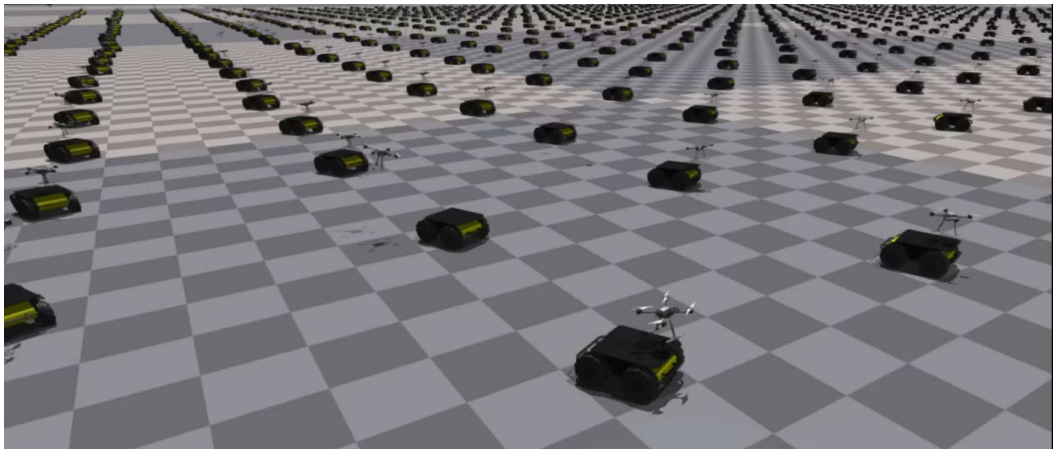

**Figure 5.** The simulation workspace in Isaac Gym shows 4098 agents training simultaneously. Each environment consists of a UAV and a Husky UGV and is separate. One environment cannot interact with another.

At the commencement of each episode, the initial positions of the agent and the moving platform were randomized. The UAV was positioned within a five-meter radius of the UGV. To add variability, the UGV's trajectory was altered at the beginning of each episode, with the UGV following one of thirty predefined trajectories at speeds reaching up to 2.2 mph. A sample landing trajectory is shown in Figure 6. During testing, the UGV was set on random trajectories that were not encountered during the training phase. The episode would reset if the UAV strayed beyond an eight-meter threshold from the moving platform

or if it made contact with the ground. Of importance, no interactive communication or coordinated behavior occurred between the Husky and the UAV during landing.

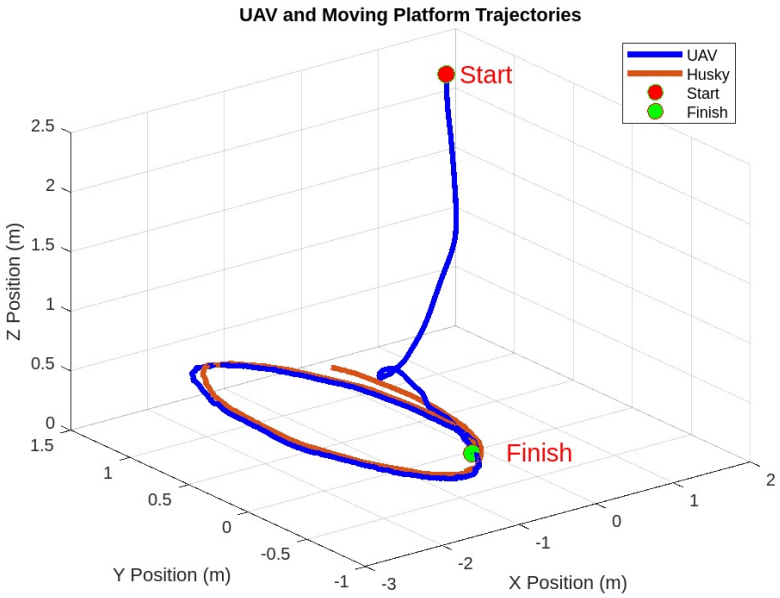


**Figure 6.** Trajectory of UAV landing on moving platform.

During the early training epochs, the UAV agents could not fly and crashed to the ground. However, as training progressed, the agents learned to track the moving platform. The UAVs crashed into the moving platform when landing, so an orientation parameter was added to the reward function to address the issue. After the training epochs, the UAV tracked and landed on the moving platform consistently. The training rewards of the fully observed states are shown in Figure 7. In the MDP environment, the training performance of the RPO-LSTM algorithm is comparable to that of the state-of-the-art PPO. After training, the agents were retrained with various partial observabilities, with each POMDP trained for 500 iterations. The simulation was run on a desktop PC with an Intel i7-11700K CPU, NVIDIA RTX 3060, and 64 GB of RAM.

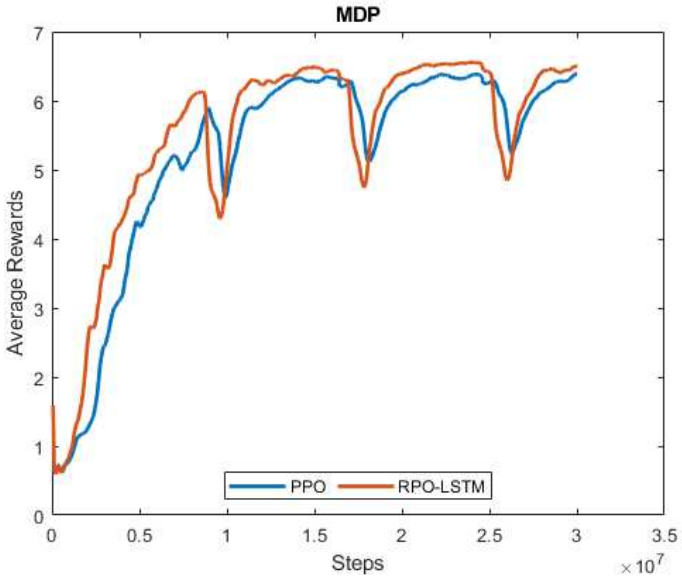


**Figure 7.** Average training rewards for the MDP scenario with no partial observability. It is observed that RPO-LSTM is able to perform as well as PPO in the MDP scenario.

### *4.3. Performance Metrics*

We utilized two metrics, namely, the average training reward and successful landing rate, to assess the effectiveness of the algorithms. The evaluation framework encompasses both learning progress and task accomplishment, thereby providing comprehensive insights into the algorithms' performance.

1. **Average Training Reward.** Employing an average training reward is a prevalent approach to evaluating machine-learning algorithms, particularly in RL contexts. The metric serves as a performance gauge, reflecting how effectively an agent learns and evolves while interacting with the environment. In our context, we computed the average rewards by evaluating Equation (4) at intervals of 16 steps.
2. **Successful Landing Metric.** Incorporating a successful landing as a performance indicator establishes a pragmatic and direct assessment of algorithm effectiveness in executing the pivotal task. A successful landing is defined as follows:

$$\mathrm{SL} = \begin{cases} 1, & \text{if distance between UAV and UGV is} \\ & \text{below 20 cm and contact is made} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

The metric is calculated in 100 trials for each POMDP method. The dual metrics offer a comprehensive evaluation framework encapsulating the learning trajectory through rewards and ultimate fulfillment of the task through successful landings.

### *4.4. Results*

This section presents a thorough evaluation of our proposed RPO-LSTM algorithm, tested across a spectrum of scenarios and measured against predefined metrics. We assessed our method against the established PPO algorithm, a benchmark in robotic RL, and a traditional approach that integrates the geometric Lee controller [40] with an EKF. Figure 8 illustrates the comparative training rewards between our RPO-LSTM and the conventional PPO within POMDP settings. Additionally, Table 2 details the successful landing rates of the PPO, Lee-EKF, and RPO-LSTM algorithms under various POMDP conditions, such as flickering, noise, and their combinations. The comparative analysis across the metrics consistently demonstrates the superior performance of our RPO-LSTM algorithm, which achieved a higher success rate in landings amid diverse conditions of uncertainty.

**Table 2.** Successful landings.

| POMDP | PPO | RPO-LSTM | Lee-EKF |
|---|---|---|---|
| NO POMDP (MDP) | 100 | 100 | 100 |
| Flicker 0.3 | 82 | 100 | 37 |
| Flicker 0.4 | 41 | 100 | 18 |
| Flicker 0.5 | 19 | 92 | 15 |
| Random Noise 0.15 | 96 | 99 | 97 |
| Random Noise 0.20 | 96 | 97 | 98 |
| Random Noise 0.25 | 87 | 98 | 98 |
| Flicker 0.1 + Noise 0.15 | 99 | 99 | 54 |
| Flicker 0.1 + Noise 0.20 | 88 | 100 | 66 |
| Flicker 0.1 + Noise 0.25 | 79 | 98 | 59 |

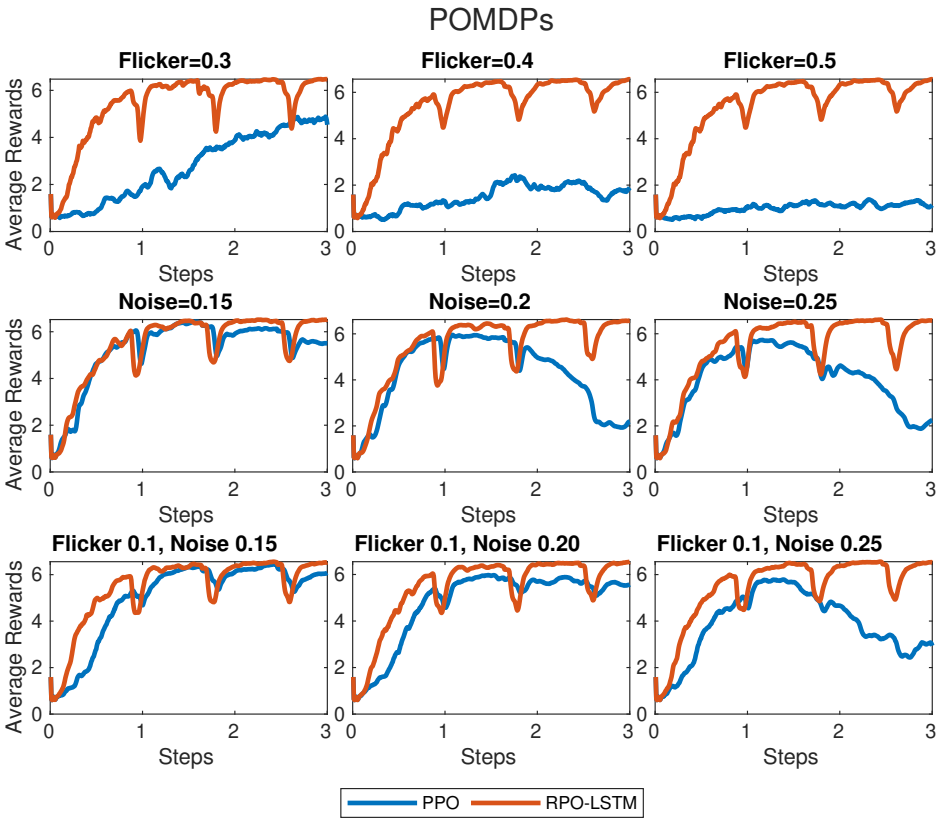


**Figure 8.** The training rewards for the task of landing the UAV on the moving platform, where, for easy comparison, only average values are plotted.

#### 4.4.1. Comparison with PPO

The most intriguing findings and behaviors emerged within the context of flickering POMDPs. Particularly, our RPO-LSTM method demonstrated a substantial performance advantage over PPO in scenarios involving flickering conditions. Notably, with up to 50% flicker, our RPO-LSTM algorithm in a POMDP environment achieved comparable performance to the MDP counterpart (i.e., nearly perfect successful landing rate as shown in Table 2), indicating robust adaptation in the presence of flicker. In the case of PPO, inference revealed that the UAV adeptly tracked and successfully landed on the moving platform under flicker conditions of up to 30%; beyond this threshold, its ability to execute successful landing maneuvers dramatically declined. Specifically, at 50% flickering, our RPO-LSTM method exhibited a better training performance with substantially higher rewards than the PPO method (Figure 8). At 50% flickering, the PPO method only managed 19 successful landings out of 100 trials, while our RPO-LSTM method still maintained a 92% successful landing rate (Table 2). Additionally, our RPO-LSTM method exhibited impressive consistency and achieved successful landings on the moving platform even under challenging conditions of up to 50% flicker.

In the context of the random noise POMDP, we intentionally introduced varying levels of noise as a means of evaluating the algorithms' performance and adaptability. The noise levels were systematically tested across a range from a 15 to 25% noise-to-signal ratio. Notably, both the PPO and RPO-LSTM algorithms showcased remarkable resilience in effectively handling observations corrupted by noise, as seen in Table 2. When disturbed by substantial levels of noise, our RPO-LSTM method exhibited a better ability to converge to policies that yielded reasonably effective outcomes than the state-of-the-art PPO method (middle row in Figure 8). The results underscore RPO-LSTM's ability to learn and adapt within the complex and noisy environments typical of the real world.

The consistent performance of both algorithms across diverse levels of random noise further accentuates their proficiency in adapting and executing successful landings in the

presence of sensory noise. While both PPO and RPO-LSTM demonstrate robust performance, an intriguing distinction is evident in the recurrent architecture of RPO-LSTM, which appears to offer an inherent advantage by enabling notably higher training rewards to be maintained(Figure 8) and success rate (Table 2) as noise levels increase. This suggests that our RPO-LSTM's ability to harness temporal dependencies through recurrent connections contributes to its stability in noisy conditions, allowing it to maintain accurate decision-making even amidst heightened sensory uncertainties.

Within the framework of the flicker-plus-noise POMDP, our experimentation involved maintaining a constant flicker level of 10% while varying the intensity of random noise between 15 and 25%. The approach enabled us to observe the effects of simultaneous disturbances of flicker and noise on the performance of the algorithms. Interestingly, the incorporation of flickers into noisy measurements noticeably impacted the landing performance of the PPO RL algorithm in comparison to scenarios involving random noise alone when the noise-to-signal ratio was above 20% (Table 2). The degradation in PPO's performance under the combination of flickers and noise suggests that the PPO algorithm's adaptability is more challenged when multiple sources of uncertainty interact.

Overall, the flicker-plus-noise POMDP scenarios underscore the adeptness of RPO-LSTM in handling complex and multi-dimensional challenges. The integration of flickers and noise did not impede its ability to execute successful landings, as seen in Table 2. This observation emphasizes the algorithm's inherent adaptability and capacity to learn in complex environments with different types of uncertainties. The unwavering success rates across these diverse scenarios reflect the algorithm's capability to navigate intricate real-world situations with great consistency. The demonstration of their potential showcases its readiness for dependable UAV autonomous landing on moving platforms under partial observability.

#### 4.4.2. Comparison with Lee-EKF

In our comparative analysis, we juxtaposed our RPO-LSTM algorithm with a classical control method, implementing a system for UAV landing on a moving platform using Lee control and an EKF, akin to the approach described in [41]. The Lee-EKF setup underwent the same simulation tests as the PPO and RPO-LSTM methods.

Under conditions of full observability, the Lee-EKF controller exhibited optimal performance that is attributed to the manually designed trajectory planner, fine-tuned for executing smooth and non-aggressive landing maneuvers.

However, the Lee-EKF's efficacy significantly deteriorated in the presence of the flicker POMDP. Vulnerability to flicker effects stems from the controller's inability to discern and disregard erroneous data. Consequently, when flicker events reset state readings to zero, the Lee-EKF erroneously acts on these nullified observations instead of dismissing them. In contrast, the RPO-LSTM demonstrated enhanced resilience in such POMDP scenarios. Through the process of reinforcement learning, it acquires the capability to identify and discard sensor data recognized as spurious. Under the flicker POMDP, the Lee-EKF managed to achieve successful landings primarily in instances when the UAV was initialized in proximity to the moving target.

As depicted in Table 2, the performance of the Lee-EKF controller closely mirrors that of our RPO-LSTM algorithm under noisy conditions. The robustness of the Lee-EKF to noise interference is largely due to the EKF's inherent noise-filtering capabilities, evidenced by the minimal impact on performance when the noise-to-signal ratio was increased from 15 to 25%, a testament to the EKF's effectiveness. Both the Lee-EKF and RPO-LSTM operated consistently despite increased noise levels. The instances of failed landings primarily occurred when the UAV was initialized too close to the moving target, leading to immediate collisions. Under noisy conditions, the trajectory patterns of all tested methods remained largely unchanged compared to scenarios devoid of POMDP influences.

However, when subjected to environments compounded by flicker and noise within a POMDP framework, the RPO-LSTM algorithm's superiority becomes evident. The Lee-

EKF's performance suffered significantly under such conditions because of its heightened susceptibility to flicker disturbances.

## 5. Ablation Study

In this section, we perform an ablation study to isolate the impact of key components within our RPO-LSTM architecture. Specifically, we investigate the effects of (1) enhanced exploration strategies (RPO), (2) the asymmetric actor–critic information flow (RPO+Critic), and (3) the integration of memory (RPO+LSTM). RPO+LSM indicates the integration of the LSTM network with RPO without the assymetric critic. For the ablation study, our RPO-LSTM is referred to as RPO-Full to indicate a combination of all the components. The study spans three POMDP scenarios—flicker in more than 100 maneuvers at 30%, random noise at 25%, and their combination. We present the outcomes in terms of maximum training rewards and successful landing rates in Tables 3 and 4. Crucially, a consistent random seed is used for all ablations in each POMDP scenario to guarantee that performance variations are solely attributable to the specific architectural components tested rather than differences in the observed scenarios.

**Table 3.** Ablation: max training rewards.

| POMDP | PPO | RPO | RPO+Critic | RPO+LSTM | RPO-Full |
|---|---|---|---|---|---|
| Flicker 0.3 | $4.88 \pm 1.4$ | $4.23 \pm 1.2$ | $\mathbf{6.54 \pm 1.6}$ | $6.48 \pm 1.5$ | $6.50 \pm 1.5$ |
| Random Noise 0.25 | $5.73 \pm 1.4$ | $6.05 \pm 1.4$ | $6.56 \pm 1.5$ | $6.45 \pm 1.4$ | $\mathbf{6.62 \pm 1.5}$ |
| Flicker 0.1 + Noise 0.25 | $5.78 \pm 1.4$ | $5.88 \pm 1.7$ | $\mathbf{6.66 \pm 1.5}$ | $6.45 \pm 1.5$ | $6.55 \pm 1.5$ |

**Table 4.** Ablation: successful landings.

| POMDP | PPO | RPO | RPO+Critic | RPO+LSTM | RPO-Full |
|---|---|---|---|---|---|
| Flicker 0.3 | 82 | 81 | 80 | 92 | 100 |
| Random Noise 0.25 | 87 | 92 | 87 | 94 | 98 |
| Flicker 0.1 + Noise 0.25 | 79 | 92 | 96 | 83 | 98 |

### *5.1. Effect of Exploration*

RPO shows an increase in performance when compared to PPO in terms of both maximum training rewards and successful landings, which is attributed to RPO's enhanced exploration capabilities. Examining the flight trajectories reveals that the RPO's trajectory closely mirrors that of the PPO. Notably, both algorithms exhibit a pronounced aggressiveness in their initial descent toward the moving platform. The similarity in trajectory suggests that the primary distinction in performance arises from RPO's ability to handle uncertainties better during exploration, rather than differences in the fundamental trajectory planning.

### *5.2. Effect of Asymmetric Actor–Critic*

The ablation study reveals compelling insights regarding the differential impacts of providing the critic with full state information while limiting the actor to partial states. The approach resulted in significantly elevated training rewards, suggesting an efficient learning process during the training phase. However, the improvement was marginal when compared to the PPO, and it underperformed relative to the RPO in terms of successful landings. The discrepancy hints at a potential issue of overfitting, where the model is too specialized to the training environment, compromising its generalizability to new or varied conditions. Notably, a majority of unsuccessful landings occurred at the onset of the maneuver, indicating that the system's handling of initial data loss was not as effective as that of other components in the architecture.

### *5.3. Effect of Adding Memory*

The ablation study's findings indicate that the inclusion of a memory component distinctly impacted the on performance metrics. Specifically, the training rewards of the

RPO with memory were intermediate, falling between those of RPO+Critic and the full RPO-Full configuration. However, in terms of successful landings, RPO+LSTM surpassed RPO+Critic, suggesting an advantage in operational effectiveness. Contrary to expectations, RPO+LSTM's performance was notably inferior to both RPO and RPO+Critic in scenarios combining flicker and noise. Interestingly, the flight trajectory associated with RPO+LSTM was observed to be the smoothest, indicating a less aggressive approach to the landing maneuver compared to the complete RPO-Full algorithm. The smoother trajectory implies a more cautious strategy, potentially prioritizing stability over the speed and assertiveness seen in other configurations.

## 6. Conclusions and Future Works

In this paper, we address the challenging task of landing a UAV on a dynamic target under partial observabilities. To solve this task, we developed an end-to-end RPO-LSTM reinforcement learning framework adept at managing incomplete sensor data and environmental noise through the utilization of temporal patterns and memory mechanisms. Validated in the high-fidelity simulation environment of Isaac Gym, our method demonstrated superior performance in executing landings on moving targets under conditions of partial observability. The RPO-LSTM algorithm consistently surpassed both the PPO learning-based method and the conventional Lee-EKF approach. In flicker scenarios, RPO-LSTM achieved up to 74% more successful landings than Lee-EKF and 50% more than PPO. In noisy environments, while all methods were effective, RPO-LSTM still outperformed PPO by 7% and Lee-EKF by 1%. In the compounded challenge of flicker and noise, RPO-LSTM maintained robust performance, exceeding PPO and Lee-EKF by as much as 11% and 41%, respectively.

In future work, we aim to transition our framework from simulation to real-world applications. Considering the demonstrated superiority of transformers over RNNs across various tasks, our future endeavors will focus on developing a robust reinforcement learning algorithm that harnesses the power of transformers to address this complex problem.

**Author Contributions:** Conceptualization, G.A. and S.J.; methodology, G.A. and K.-D.N.; simulation, G.A. and S.J.; data analysis and interpretation, G.A. and K.-D.N.; manuscript preparation, G.A. and K.-D.N. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported by the National Science Foundation under Grants #2245022 and #2138206.

**Data Availability Statement:** The data that support the findings of this study are available from the corresponding author upon reasonable request.

**Acknowledgments:** We would like to express our sincere gratitude to Omar Qasem for his valuable assistance in reviewing our experiments and revising our paper. His insightful comments and suggestions greatly improved the quality and clarity of our work.

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

## References

1. Doitsidis, L.; Weiss, S.; Renzaglia, A.; Achtelik, M.W.; Kosmatopoulos, E.; Siegwart, R.; Scaramuzza, D. Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. *Auton. Robot.* **2012**, *33*, 173–188. [CrossRef]
2. Cherubini, A.; Papini, A.; Vertechy, R.; Fontana, M. Airborne Wind Energy Systems: A review of the technologies. *Renew. Sustain. Energy Rev.* **2015**, *51*, 1461–1476. [CrossRef]
3. Williams, A.; Yakimenko, O. Persistent mobile aerial surveillance platform using intelligent battery health management and drone swapping. In Proceedings of the 2018 4th International Conference on Control, Automation and Robotics (ICCAR), Auckland, New Zealand, 20–23 April 2018; pp. 237–246.
4. Scott, J.; Scott, C. Drone delivery models for healthcare. In Proceedings of the 50th Hawaii International Conference on System Sciences, Hilton Waikoloa Village, HI, USA, 4–7 January 2017.

5. Arora, S.; Jain, S.; Scherer, S.; Nuske, S.; Chamberlain, L.; Singh, S. Infrastructure-free shipdeck tracking for autonomous landing. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 323–330.
6. Hu, B.; Lu, L.; Mishra, S. Fast, safe and precise landing of a quadrotor on an oscillating platform. In Proceedings of the 2015 American Control Conference (ACC), Chicago, IL, USA, 1–3 July 2015; pp. 3836–3841.
7. Falanga, D.; Zanchettin, A.; Simovic, A.; Delmerico, J.; Scaramuzza, D. Vision-based autonomous quadrotor landing on a moving platform. In Proceedings of the 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, China, 11–13 October 2017; pp. 200–207.
8. Liu, X.; Zhang, S.; Tian, J.; Liu, L. An onboard vision-based system for autonomous landing of a low-cost quadrotor on a novel landing pad. *Sensors* **2019**, *19*, 4703. [CrossRef] [PubMed]
9. Keipour, A.; Pereira, G.A.S.; Bonatti, R.; Garg, R.; Rastogi, P.; Dubey, G.; Scherer, S. Visual Servoing Approach to Autonomous UAV Landing on a Moving Vehicle. *Sensors* **2022**, *22*, 6549. [CrossRef] [PubMed]
10. Fourlas, G.K.; Karras, G.C. A Survey on Fault Diagnosis and Fault-Tolerant Control Methods for Unmanned Aerial Vehicles. *Machines* **2021**, *9*, 197. [CrossRef]
11. Rodriguez-Ramos, A.; Sampedro, C.; Bavle, H.; De La Puente, P.; Campoy, P. A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. *J. Intell. Robot. Syst.* **2019**, *93*, 351–366. [CrossRef]
12. Jung, W.; Kim, Y.; Bang, H. Target state estimation for vision-based landing on a moving ground target. In Proceedings of the 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 7–10 June 2016; pp. 657–663. [CrossRef]
13. Keller, A.; Ben-Moshe, B. A Robust and Accurate Landing Methodology for Drones on Moving Targets. *Drones* **2022**, *6*, 98. [CrossRef]
14. Xu, L.; Luo, H. Towards autonomous tracking and landing on moving target. In Proceedings of the 2016 IEEE International Conference on Real-Time Computing and Robotics (RCAR), Angkor Wat, Cambodia, 6–10 June 2016; pp. 620–628. [CrossRef]
15. Serra, P.; Cunha, R.; Hamel, T.; Cabecinhas, D.; Silvestre, C. Landing of a Quadrotor on a Moving Target Using Dynamic Image-Based Visual Servo Control. *IEEE Trans. Robot.* **2016**, *32*, 1524–1535. [CrossRef]
16. Hu, B.; Mishra, S. Time-Optimal Trajectory Generation for Landing a Quadrotor Onto a Moving Platform. *IEEE/ASME Trans. Mechatronics* **2019**, *24*, 585–596. [CrossRef]
17. Paris, A.; Lopez, B.T.; How, J.P. Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 9577–9583. [CrossRef]
18. Xia, K.; Huang, Y.; Zou, Y.; Zuo, Z. Reinforcement Learning Control for Moving Target Landing of VTOL UAVs with Motion Constraints. *IEEE Trans. Ind. Electron.* **2024**, *71*, 7735–7744. [CrossRef]
19. Jiang, Y.; Li, C.; Dai, W.; Zou, J.; Xiong, H. Monotonic robust policy optimization with model discrepancy. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 4951–4960.
20. Mankowitz, D.J.; Levine, N.; Jeong, R.; Shi, Y.; Kay, J.; Abdolmaleki, A.; Springenberg, J.T.; Mann, T.; Hester, T.; Riedmiller, M. Robust reinforcement learning for continuous control with model misspecification. *arXiv* **2019**, arXiv:1906.07516.
21. Qasem, O.; Gao, W. Robust Policy Iteration of Uncertain Interconnected Systems with Imperfect Data. *IEEE Trans. Autom. Sci. Eng.* **2023**, *21*, 1214–1222. [CrossRef]
22. Meng, L.; Gorbet, R.; Kulić, D. Memory-based deep reinforcement learning for POMDPs. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp 5619–5626.
23. Wang, Y.; He, H.; Tan, X. Robust reinforcement learning in POMDPs with incomplete and noisy observations. *arXiv* **2019**, arXiv:1902.05795.
24. Aikins, G.; Jagtap, S.; Gao, W. Resilience Analysis of Deep Q-Learning Algorithms in Driving Simulations Against Cyberattacks. In Proceedings of the 2022 1st International Conference On AI In Cybersecurity (ICAIC), Houston, TX, USA, 24–26 May 2022; pp. 1–6.
25. Hickling, T.; Aouf, N.; Spencer, P. Robust Adversarial Attacks Detection based on Explainable Deep Reinforcement Learning for UAV Guidance and Planning. *IEEE Trans. Intell. Veh.* **2023**, *8*, 4381–4394. [CrossRef]
26. Gleave, A.; Dennis, M.; Wild, C.; Kant, N.; Levine, S.; Russell, S. Adversarial Policies: Attacking Deep Reinforcement Learning. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
27. Saj, V.; Lee, B.; Kalathil, D.; Benedict, M. Robust Reinforcement Learning Algorithm for Vision-based Ship Landing of UAVs. *arXiv* **2022**, arXiv:2209.08381.
28. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
29. Rahman, M.M.; Xue, Y. Robust Policy Optimization in Deep Reinforcement Learning. *arXiv* **2022**, arXiv:2212.07536.
30. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
31. Chikhaoui, K.; Ghazzai, H.; Massoud, Y. PPO-based Reinforcement Learning for UAV Navigation in Urban Environments. In Proceedings of the 2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS), Fukuoka, Japan, 7–10 August 2022; pp. 1–4.

32. Piponidis, M.; Aristodemou, P.; Theocharides, T. Towards a Fully Autonomous UAV Controller for Moving Platform Detection and Landing. In Proceedings of the 2022 35th International Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID), Bangalore, India, 26 February–2 March 2022; pp. 180–185.
33. Ladosz, P.; Mammadov, M.; Shin, H.; Shin, W.; Oh, H. Autonomous Landing on a Moving Platform Using Vision-Based Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2024**, *9*, 4575–4582. [CrossRef]
34. Feng, Y.; Zhang, C.; Baek, S.; Rawashdeh, S.; Mohammadi, A. Autonomous Landing of a UAV on a Moving Platform Using Model Predictive Control. *Drones* **2018**, *2*, 34. [CrossRef]
35. Pinto, L.; Andrychowicz, M.; Welinder, P.; Zaremba, W.; Abbeel, P. Asymmetric Actor Critic for Image-Based Robot Learning. In Proceedings of the 14th Robotics: Science and Systems, RSS 2018, Pittsburgh, PA, USA, 26–30 June 2018; Kress-Gazit, H., Srinivasa, S., Howard, T., Atanasov, N., Eds.; MIT Press Journals: Cambridge, MA, USA , 2018. [CrossRef]
36. Nahrendra, I.M.A.; Tirtawardhana, C.; Yu, B.; Lee, E.M.; Myung, H. Retro-RL: Reinforcing Nominal Controller with Deep Reinforcement Learning for Tilting-Rotor Drones. *IEEE Robot. Autom. Lett.* **2022**, *7*, 9004–9011. [CrossRef]
37. Urain, J.; Funk, N.; Peters, J.; Chalvatzaki, G. SE(3)-DiffusionFields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 5923–5930. [CrossRef]
38. Margolis, G.B.; Agrawal, P. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In Proceedings of the Conference on Robot Learning, Atlanta, GA, USA, 6–9 November 2023; pp. 22–31.
39. Radosavovic, I.; Xiao, T.; Zhang, B.; Darrell, T.; Malik, J.; Sreenath, K. Real-world humanoid locomotion with reinforcement learning. *Sci. Robot.* **2024**, *9*, eadi9579. [CrossRef] [PubMed]
40. Lee, T.; Leok, M.; McClamroch, N.H. Geometric tracking control of a quadrotor UAV on SE(3). In Proceedings of the 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, 15–17 December 2010; pp. 5420–5425. [CrossRef]
41. Araar, O.; Aouf, N.; Vitanov, I. Vision based autonomous landing of multirotor UAV on moving platform. *J. Intell. Robot. Syst.* **2017**, *85*, 369–384. [CrossRef]


**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.