drones

MDPI

Article

Deep Reinforcement Learning with Corrective Feedback for Autonomous UAV Landing on a Mobile Platform

Lizhen Wu [1], Chang Wang [1,*], Pengpeng Zhang [2] and Changyun Wei [2]

[1] College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
[2] College of Mechanical and Electrical Engineering, Hohai University, Changzhou 213022, China
* Correspondence: wangchang07@nudt.edu.cn

Abstract: Autonomous Unmanned Aerial Vehicle (UAV) landing remains a challenge in uncertain environments, e.g., landing on a mobile ground platform such as an Unmanned Ground Vehicle (UGV) without knowing its motion dynamics. A traditional PID (Proportional, Integral, Derivative) controller is a choice for the UAV landing task, but it suffers the problem of manual parameter tuning, which becomes intractable if the initial landing condition changes or the mobile platform keeps moving. In this paper, we design a novel learning-based controller that integrates a standard PID module with a deep reinforcement learning module, which can automatically optimize the PID parameters for velocity control. In addition, corrective feedback based on heuristics of parameter tuning can speed up the learning process compared with traditional DRL algorithms that are typically time-consuming. In addition, the learned policy makes the UAV landing smooth and fast by allowing the UAV to adjust its speed adaptively according to the dynamics of the environment. We demonstrate the effectiveness of the proposed algorithm in a variety of quadrotor UAV landing tasks with both static and dynamic environmental settings.

Keywords: deep reinforcement learning; interactive learning; DDPG; PID; UAV landing

check for updates

Citation: Wu, L.; Wang, C.; Zhang, P.; Wei, C. Deep Reinforcement Learning with Corrective Feedback for Autonomous UAV Landing on a Mobile Platform. *Drones* **2022**, *6*, 238. https://doi.org/10.3390/drones6090238

Academic Editors: Yu-Jun Zheng and Mumtaz Karatas

Received: 17 August 2022
Accepted: 1 September 2022
Published: 4 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Unmanned Aerial Vehicles (UAVs) have been widely used in a variety of real-world applications, such as civil engineering [1], precision agriculture [2], and monitoring in mining areas [3]. One advantage of using UAVs is that they can fly to and land on complex terrains that are more difficult to reach through the ground traverse. However, UAVs have drawbacks of relatively short flight time and low load limit compared with ground platforms such as Unmanned Ground Vehicles (UGVs). Alternatively, collaborating UAVs and UGVs is a more efficient and effective way to solve complex field tasks [4]. On the one hand, UAVs can fly up to a certain height and provide a global map that aids UGVs in planning and choosing the nearest path to the destination. On the other hand, UGVs can provide UAVs with charging facilities that guarantee the flight time as needed.

However, the autonomous landing of a UAV on a UGV is still challenging, as discussed in [5]. Specifically, the motion dynamics of the UGV are unknown for the UAV that has to perform the landing task with high uncertainty. To solve the landing problem, a variety of methods have been proposed, such as fuzzy control [6], Model Predictive Control (MPC) [7], PD (Proportional, Derivative) [8] control, PID (Proportional, Integral, Derivative) control [9], vision-based control [10] together with reinforcement learning [11–14].

Some of the approaches only considered UAV landing on static platforms [6,8] or in simulation [7]. A basic PID controller was used to design a collaborative UGV-UAV system for data collection in the application of the construction industry [9]. One drawback of the PID controller is that a fixed gain cannot provide an immediate response to overcome the nonlinear thrust effect with decreasing altitude. In addition, the parameters of

traditional PID controllers are all constant numbers that need manual tuning. Therefore, such controllers can hardly handle dynamic situations such as landing with various initial conditions or landing on a moving platform.

Alternatively, learning-based methods have been integrated with the traditional PID controller for solving tasks in dynamic environments. Specifically, Reinforcement Learning (RL) has become popular and has been combined with PID to improve the accuracy of path planning for mobile robots [15,16]. The results of combining Q-learning [17] and PID have proved better than Q-learning or PID alone. However, tabular Q-learning requires discrete states and actions that can hardly handle high-dimensional or continuous control problems in many real-world tasks. Recently, more advanced Deep Reinforcement Learning (DRL) algorithms, such as Deep Deterministic Policy Gradient (DDPG) [18], Proximal Policy Optimization (PPO) [19], and Soft Actor-Critic (SAC) [20], can output continuous actions based on high-dimensional sensory input. Specifically, DDPG was found effective in handling disturbances for vision-based UAV landing [12]. The controller was trained in simulation and transferred to a real-world environment, but the output would be the same even if the heights were different because the altitude (z-direction) was not considered in the state representation. Another work [14] solved this problem by considering three-dimensional directions in the state representation and also chose DDPG for vision-based UAV landing on a moving platform. However, these methods suffer the same problem as most deep reinforcement learning algorithms that rely on heavy offline training with high-quality samples.

Corrective feedback from a human teacher can possibly speed up the learning process if the teacher has a good understanding of the task as well as the dynamics of the environment. The DAGGER method required the human expert to label each queried state visited by the learner [21]. HG-DAGGER reduced the alertness burden on the expert by executing a human-gated mixed control trajectory and using the human-labeled portions of the data as the online batch update [22]. In a more natural and efficient manner, the EIL approach made use of non-intervention in addition to the intervention of human feedback [23]. In another work, the TAMER framework allowed a human to interactively shape an agent's policy via evaluative feedback [24]. The credit assignment mechanism associated the feedback with the relevant data of state-action pairs. Based on the structure of TAMER, the COACH framework advocated using the feedback in the action domains, and past feedback was considered for adjusting the amount of human feedback that a given action received adaptively [25]. Furthermore, corrective feedback was used to construct action exploration strategies in continuous spaces [26,27]. However, the human teacher would not always be able to give appropriate feedback for problems with fast and complex transitions in high-dimensional action spaces, e.g., learning to control a UAV landing on a mobile platform. In that case, the learning curve would be similar to pure reinforcement learning since few feedback signals would be given by the teacher [26].

In this paper, we solve the quadrotor UAV landing task in dynamic environments using the PID controller combined with deep reinforcement learning as well as corrective feedback based on heuristics. Similar to a recent study [28] that uses an adaptive learning navigation rule for UAV landing on a moving vehicle, the heuristics in this paper are in terms of rules based on the experience of a human expert. We note that there are many choices of reinforcement learning algorithms that can handle high-dimensional states and continuous actions, and we choose the DDPG algorithm without the loss of generality. As a result, our method can automatically learn the optimal parameters of the PID controller so that the human operator can be relieved from the heavy workload of manual parameter tuning of the PID controller. Compared with the previous work [15,16], our method has better generalization capability for landing with uncertain initial conditions, as well as landing with reliable performance on mobile ground platforms. In addition, our method has the advantage of high efficiency over the vision-based deep reinforcement learning methods [12,14] due to the use of heuristics for parameter tuning, which speeds up the learning process with immediate feedback rather than waiting for sparse rewards, as in

many RL algorithms. Different to the interactive learning literature in which a human typically intervenes occasionally [21,23–25], our corrective feedback is available at every time step if needed for the PID controller.

From the perspective of designing an intelligent control system with respect to human–computer interaction, the main innovation of our work is that we have decoupled the UAV landing control problem using a hierarchical framework. Specifically, a low-level PID controller is responsible for providing fast reactive signals to control the speed of the upward rotors, while a high-level agent or human corrective feedback does not need to pay attention to the rotor control. However, the PID controller is known for its difficult parameter tuning issue, and human designers are usually needed to fine-tune the PID gains, which is a time-consuming and challenging task for the risky UAV landing problem. In this work, the gains would be adapted by the high-level learning agent if the operation conditions were changed. To achieve the fine-tuning of the PID gains, the agent does not need to learn from scratch, as the human corrective feedback can regulate the agent's action selection. On the one hand, the human's knowledge about the landing task can be incorporated before the task starts to improve the safety of the UAV. On the other hand, the real-time feedback from a human can accelerate the convergence of the task learning process.

The remainder of the paper is organized as follows. Section 2 briefly introduces reinforcement learning. Section 3 proposes our approach, followed by experiments and results in Section 4. Finally, we conclude the paper in Section 5.

2. Preliminaries

2.1. Reinforcement Learning

A Reinforcement Learning (RL) agent manages to find optimal actions in given states by maximizing the expected accumulated rewards through trial-and-error interaction with the environment. Typically, an RL problem can be described by five elements S, A, P, r and γ, where S denotes the state space and a specific state $s \in S$, A denotes the action space and an action $a \in A$, r denotes the reward function, and R_t stands for the accumulated reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$ received from the time step t to T. P represents the state-transition model, and γ is a discount factor.

The state-value function V^π of a state s_t following a policy π is defined as the expected accumulated reward as follows

$$V^\pi(s_t) = \mathbb{E}[R_t | s = s_t, \pi]. \tag{1}$$

Similarly, the action-value function Q^π of (s_t, a_t) following a policy π is defined as follows

$$Q^\pi(s_t, a_t) = \mathbb{E}[R_t | s = s_t, a = a_t, \pi]. \tag{2}$$

The expected reward $J(\pi)$ is an evaluation function of a policy π defined as follows

$$J(\pi) = \mathbb{E}[R_t | \pi]. \tag{3}$$

The optimal policy $\pi^*(a_t|s_t)$ means that an optimal action a^* would be selected in the state s_t, which maximizes the Q-value function as follows

$$a^* = \arg\max_{a_t} Q^\pi(s_t, a_t). \tag{4}$$

Q-learning [17] is a popular algorithm for finding the optimal action selection policy for discrete states and actions. Based on Q-learning, a variety of algorithms such as DQN [29], double DQN [30] and dueling DQN [30] have been proved effective in solving high-dimensional problems.

2.2. Deep Reinforcement Learning

Many real-world control problems have to be solved in continuous state and action spaces. Function approximators have been used to represent the state-value and action-value functions, trying to alleviate the issue of the curse of dimensionality. Neural networks have become a popular choice of function approximators, especially due to the power of deep neural networks such as CNN. Accordingly, we can optimize the parameters θ^Q of a neural network by a loss function as follows:

$$L(\theta^Q) = \mathbb{E}[(y_t - Q(s_t, a_t|\theta^Q))^2], \tag{5}$$

where $y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}|\theta^Q)$. If π is an arbitrary deterministic policy, we describe it as a mapping from states to actions $\mu : \mathrm{S} \rightarrow \mathrm{A}$ and omit the expectation:

$$Q^\mu(s_t, a_t) = r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1})). \tag{6}$$

Then, we define an actor function $\mu(s|\theta^\mu)$ as a mapping from every state to a particular action. The actor function represented by a neural network is updated based on the expected return $J(\pi)$ as follows

$$\begin{aligned} \nabla_{\theta^\mu} J(\pi) &\approx \mathbb{E}[\nabla_{\theta^\mu} Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i|\theta^\mu)}] \\ &= \mathbb{E}[\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}]. \end{aligned} \tag{7}$$

The DDPG [18] algorithm concurrently learns a Q-function and a policy using two neural networks, one for the actor and one for the critic. The actor network takes the current state as input and an action as output in the continuous action space. The critic evaluates the current state and action of the actor by calculating the corresponding Q-value. However, simultaneously updating the two neural networks is unstable and can cause divergence. Another two target networks, for both the actor and the critic, are employed to generate the targets for computing the Time Difference (TD) errors for the learning. As a result, the stability of the algorithm is increased.

The target networks have the same structures as the two actor and critic networks. In practice, a random disturbance is added to every action for exploration. After each action execution, the transition (s_{step},a_{step},r_{step},s_{step+1}) is stored in a replay buffer. The critic network is updated based on Equation (8) when the replay buffer is full, where B is the size of a sampled batch,

$$L = \left(\frac{1}{B}\right) * \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2. \tag{8}$$

For every step, the actor network is updated as follows

$$\nabla_{\theta^\mu} J \approx \frac{1}{B} \sum_i \nabla_a Q(s,\ a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{\cdot s_i} \tag{9}$$

Then, we can update the target networks,

$$\theta^{Q'} = \tau\theta^Q + (1-\tau)\theta^{Q'}, \theta^{\mu'} = \tau\theta^\mu + (1-\tau)\theta^{\mu'}. \tag{10}$$

After training with sufficient episodes, the converged target networks can be used to solve the problems.

2.3. Reinforcement Learning with Corrective Feedback

As an RL agent typically requires trial-and-error interactions with the environment to collect sufficient experiences so as to optimize its control policy, learning from scratch requires exploring the entire state and action spaces, which can take quite some time.

Similar to the interactive learning framework COACH [25], we use corrective feedback in terms of a binary signal, i.e., to increase or decrease the action selected by the RL agent, to speed up the RL process (see Figure 1).

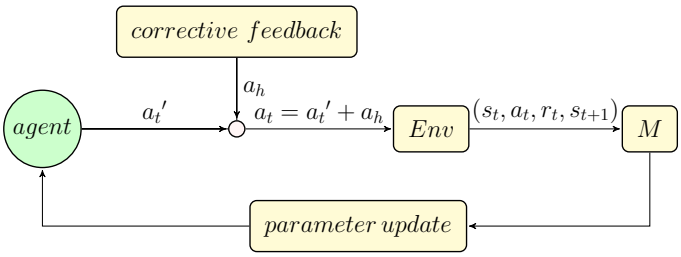


Figure 1. RL with corrective feedback based on the human experience of the task.

The corrective feedback serves as a guidance for action selection during reinforcement learning. In other words, the agent selects an action a_t', and the feedback of bias a_h would be added to decide a final action a_t. It is expected that the human has a better understanding of how well the task is performed and, therefore, can provide an immediate positive or negative reward to generate an appropriate action advice a_h towards the optimal action, as shown in the literature [25–27]. However, the human advice can not be guaranteed to be always correct or accurately associated with the situations to be improved. In contrast, we design the corrective feedback as a module of heuristic rules that define when and how the actions should be biased (see Section 3.4). After the action a_t is performed, the agent observes a reward r_t and a new state s_{t+1}, and the data (s_t,a_t,r_t,s_{t+1}) is saved in a memory buffer M. Then, M can be used by a deep reinforcement learning algorithm using the experience replay mechanism. Typically, the function approximation technique can be used to represent the actor and critic. If M is full, the critic and actor networks are updated one time per episode. The parameters are updated in an online learning fashion.

3. Approach

In this section, we first introduce the UAV dynamics and present how we use the standard PID controller for UAV landing. Then, we explain how to combine PID with RL. Finally, we modify the learning-aided PID control with corrective feedback.

3.1. Uav Dynamics

In order for the UAV to land on the ground vehicle, the UAV estimates its relative position to the landing platform using a camera installed underneath the UAV. We use the North East Down (NED) frame and a body frame to describe the UAV landing process. Since the UAV is a rigid body, the NED frame $\{o_e, x_e, y_e, z_e\}$ is the inertia frame based on the earth, and o_e denotes the center of the earth. The body frame $\{o_b, x_b, y_b, z_b\}$ is attached to the UAV fuselage, and o_b indicates the mass center of the UAV.

To describe the rotational motion, we define the rotation matrix $R \in \mathbb{R}^{3\times3}$ and the Euler angles $[\phi, \theta, \psi]^T$ that represent the pitch, raw and yaw, respectively. The rotation matrix can be obtained based on the Euler anglers,

$$R = \begin{bmatrix} c\theta c\psi & s\theta c\psi s\phi - s\psi c\phi & s\theta c\psi c\phi + s\psi s\phi \\ c\theta s\psi & s\theta s\psi s\phi + c\psi c\phi & s\theta s\psi c\phi - c\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}, \tag{11}$$

where the operators s and c denote $\sin(\cdot)$ and $\cos(\cdot)$ for the sake of simplicity. The kinematics of the UAV can be described as follows

$$\begin{cases} \dot{p} = v \\ \dot{v} = -ge_3 + \frac{T}{m}Re_3 + \frac{T_d}{m} \\ W\dot{\gamma} = \omega \\ J\dot{\omega} = -\omega \times J\omega + \tau + \tau_d. \end{cases} \tag{12}$$

Here we use $p = \left[p_x, p_y, p_z\right]^T$, $v = \left[v_x, v_y, v_z\right]^T$, and $\omega = \left[\omega_x, \omega_y, \omega_z\right]^T$ to denote the position, the linear velocity, and the angular velocity of the UAV according to the body frame, respectively. With regard to the acceleration, g indicates the local gravitational acceleration, m is the total mass, T is the applied thrust along the vector $e_3 = [0, 0, 1]$, and T_d represents disturbance. The angular velocity can be calculated based on the Euler angers $\gamma = [\psi, \theta, \phi]^T$, and the attitude transition matrix W is defined as

$$W = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}. \tag{13}$$

In Equation (12), J means the inertial matrix according to the UAV body frame, and τ and τ_d represent the applied torque and the disturbance torque, respectively.

3.2. Baseline: Standard PID for UAV Landing

A PID controller provides a low-level control loop that calculates control actions based on the error signal $e(t)$, which is the deviation between the desired set-point and the current measurement. The structure of a standard PID controller is shown in Figure 2.

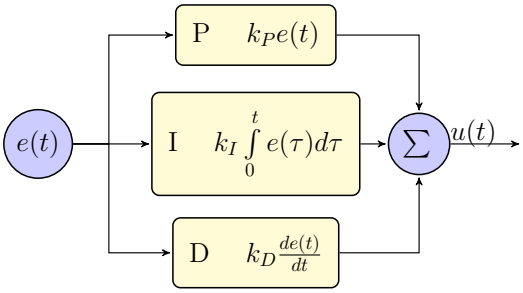


Figure 2. Standard structure of a PID controller.

The PID controller continuously corrects the output based on the three control parameters, i.e., proportional, integral and derivative gains, denoted by k_P, k_I and k_D, respectively. The three parameters are updated according to the error signal, and the control signal $u(t)$ is obtained as follows

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \frac{de(t)}{dt}. \tag{14}$$

In this work, we employ velocity control for safe landing, and thus, the commands will be sent to the UAV to adjust its velocities until it reaches the landing platform. Here the error signal $e(t)$ reflects the distance between the UAV and the centroid of the landing area, which is detected and localized using a vision-based method. The control variables are calculated based on the detected errors and the PID gains.

3.3. PID with RL for UAV Landing

The framework of PID integrated with RL is shown in Figure 3. The framework consists of two modules. The RL module is shown within the blue dashed lines, and the PID module is shown within the green dashed lines.

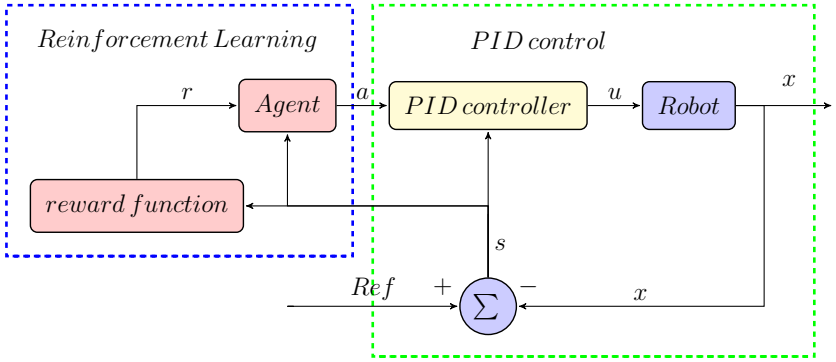


Figure 3. The framework of PID with RL.

Denote by $p_{uav} = (p_x, p_y, p_z)$ the position of UAV in the 3D world coordinate system, and $\boldsymbol{x} = (p_x, p_y, p_z)$. The reference signal Ref indicates the goal position $p_g = (p_{g_x}, p_{g_y}, p_{g_z})$ of the UAV, i.e., the horizontal surface center of the ground vehicle. The state vector $\boldsymbol{s} = (d_x, d_y, d_z)$ is three-dimensional, indicating the distances from p_{uav} to p_g in the x, y and z directions, respectively. The output of the PID controller is $\boldsymbol{u} = (v_x, v_y)$, where v_x and v_y are the velocities in the x and y directions.

The action of the agent a consists of the three PID parameters k_P, k_I, k_D that can be adjusted by the RL module at any time step if needed. In this paper, we use PID to control the velocities in the x and y directions, assuming the velocity in the vertical z direction as constant for safety reasons. In other words, the action $a = (k_P^{v_x}, k_I^{v_x}, k_D^{v_x}, k_P^{v_y}, k_I^{v_y}, k_D^{v_y})$ is six-dimensional.

The reward function r_t is defined as follows

$$r_t = \begin{cases} 1, & \textit{if successful} \\ -1, & \textit{if failed} \\ d_{t-1} - d_t, & \textit{otherwise} \end{cases}, \tag{15}$$

where d_t indicates the distance between the UAV and the goal position at the time step t. If the UAV reaches the target position and lands successfully, the reward is 1, and the episode ends. If the UAV fails, the reward is -1, and the episode also ends. Otherwise, the reward is the difference between the distance between the last time step and the current time step. We note that this reward function encourages fast landing towards the goal position and punishes fast landing away from the goal position. Due to the contribution of the RL module, the PID controller is expected to be more adaptive to changing situations.

3.4. Rl with Corrective Feedback for UAV Landing

Although the RL algorithm enables automatic parameter turning of the PID controller, the learning process is time-consuming. We assume that the human is likely to have a good understanding of how the landing task should be carried out and can therefore provide heuristics to influence the action selection of the UAV towards faster learning of the optimal landing policy.

According to the experience of the human expert, the P-gains of the PID controller have significant influence on the UAV landing task, i.e., $k_P^{v_x}$ and $k_P^{v_y}$. Higher values of $k_P^{v_x}$ and $k_P^{v_y}$ may result in a greater change in speed in the x and y directions. When the UAV is far from the goal position, i.e., the error signal $e(t)$ is high, then higher values of $k_P^{v_x}$ and $k_P^{v_y}$ are preferred for decreasing $e(t)$ faster. However, if the P-gain is too high (e.g., higher than 1.0), it might result in high velocity so that the UAV would easily lose sight of the ground vehicle. On the other hand, if the P-gain is too small (e.g., smaller than 0.2), it would have little impact on the velocity change; therefore, the P-gain needs to be increased.

We illustrate the proposed approach of PID with DDPG [18] and corrective feedback in Algorithm 1. We note that many other reinforcement learning algorithms should also work with the method illustrated in Figure 3.

Algorithm 1 PID with DDPG and corrective feedback

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ
Initialize target network $Q(s, a|\theta^{Q'})$ and $\mu(s|\theta^{\mu'})$ with weights $\theta^{Q'}$,$\theta^{\mu'}$, where $\theta^Q \rightarrow \theta^{Q'}$,$\theta^\mu \rightarrow \theta^{\mu'}$
Initialize the replay buffer M
for $episode = 1$ to N_1 **do**:
    Receive initial observation state s_t
    for $t = 1$ to N_2 **do**:
        Select a primary action $a'_t = \mu(s|\theta^\mu)$ according to the current policy
        Receive corrective feedback a_h
        Select the action $a_t = a'_t + a_h$
        Update the parameters of the PID controller with a_t
        Observe the reward r_t and the new state s_{t+1}
        Save the transition (s_t,a_t,r_t,s_{t+1}) in M
        Sample a random mini-batch of (s_i,a_i,r_i,s_{i+1}) from M
        Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s|\theta^{\mu'}|\theta^{Q'}))$
        Update the critic network using Equation (8)
        Update the actor policy using Equation (9):
        Update the target network using Equation (10)
    end for
end for

We assumed that the range of the RL agent's action a'_t was $(0, 0.6)$, and the corrective feedback was $a_h = -0.2$ or $a_h = 0.2$. As mentioned above, if the human considered that the velocity of the UAV could be increased, then $a_h = 0.2$. As a result, the UAV would accelerate towards the target position. Otherwise, if the human considered that the velocity of the UAV should be decreased, then $a_h = -0.2$. The following heuristics were used to construct the following rules of corrective feedback:

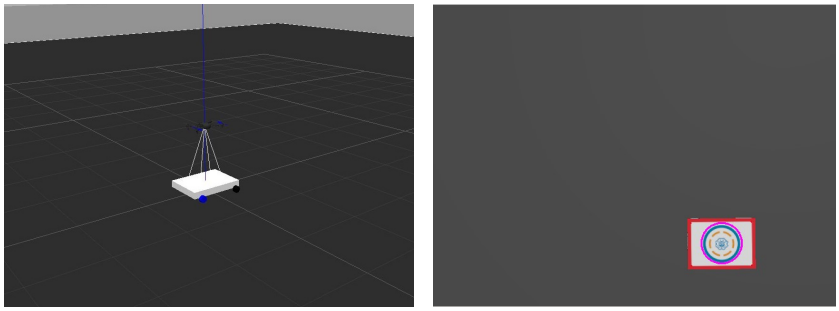
- If $k_P^{v_x} > 1$ or $k_P^{v_y} > 1$, then $a_h = -0.2$.
- If $k_P^{v_x} < 0.2$ or $k_P^{v_y} < 0.2$, then $a_h = 0.2$.

4. Experiments and Results

4.1. Environmental Settings

We first carried out the quadrotor UAV landing task in a simulated environment using the Gazebo simulator [31] (see Figure 4). The UAV was controlled by the ROS package [32]. The velocity of the UAV in the z direction was set to 0.2 m/s by default.

The ground vehicle could move forward and backward and turn at a certain angle. The size of the ground vehicle (0.6 m × 0.8 m × 0.2 m) was larger than that of the UAV (0.4 m × 0.4 m) to leave enough space for landing. We stuck a designed marker (0.6 m × 0.8 m) on top of the horizontal surface of the mobile ground vehicle. It was recognized by the UAV's downside camera for the purpose of detection and estimation. The marker had smaller circular patterns at its center, used for localization when the UAV was close to the ground vehicle.



(**a**) (**b**)

Figure 4. A quadrotor UAV landing task in the simulation environment. (**a**) Environmental setting. (**b**) Recognized marker on the mobile vehicle.

In this work, since the UAV needs to land on the ground vehicle, we develop a vision-based method to detect the landing platform. As shown in Figure 4, a designed landmark is placed on the surface of the platform for the UAV to recognize. In order to achieve a lightweight visual-based detection, the relative position of the platform is estimated based on the circle of the landmark. We first convert the RGB images captured by the UAV camera to the HSV color model so as to eliminate other colors, with the exception of the blue color. Then, the HSV mask can convert the RGB image to a grey image, and a binary image of the landmark can be obtained by threshold segmentation of the grey image. Finally, we can identify the circular feature and estimate the center (x_c, y_c) and the diameter of the detected circle. In addition, the altitude of the UAV can be calculated based on the focal length of the UAV's camera and the size of the detected circle in the UAV's camera view.

For the three learning-based approaches, i.e., RL (DDPG), PID with RL (RL-PID) and PID with RL and corrective feedback (RLC-PID), we trained the agent for 400 episodes, where $p_0 = (0, 0, 4.0)$ and $p_g = (0, 0, 0.2)$. We expect that the UAV could always keep track of the marker for localization. If the marker was out of sight, it would be considered a failure, and the UAV started a new episode. The ground vehicle was assumed static during training for the purpose of faster policy learning. Then, it was allowed to move during the testing to compare the performance of the controllers in dynamic situations. We note that the UAV hovered for a while before landing towards p_g. Due to the hovering error, the actual initial position of training was within a radius of about 0.1 m around p_0 in the three-dimensional space. The introduced uncertainty made the problem more challenging than landing from exactly the same initial position.

The parameters of DDPG were set empirically as follows. The learning rates for both the actor and critic networks were 0.0001. The target network was updated every 100 time steps. The discount factor of the reward was 0.9. The memory buffer $M = 2000$, and the mini-batch size was 64.

4.2. Training in the Simulation Environment

The success times and training time (in minutes) were compared among RL(DDPG), RL-PID and RLC-PID in Figure 5. The RL-PID method succeeded more than the RL method, and the RLC-PID method was even better, with a near 100% success rate. We note that the RL approach resulted in many failures in which the UAV lost track of the marker; therefore, it was terminated earlier and took less time than RL-PID. The required training time of RLC-PID was also the shortest. The reason is that the RL module encouraged the UAV to optimize the PID parameters for fast learning of a stable landing policy. In addition, the corrective feedback can further speed up the parameter optimization process.

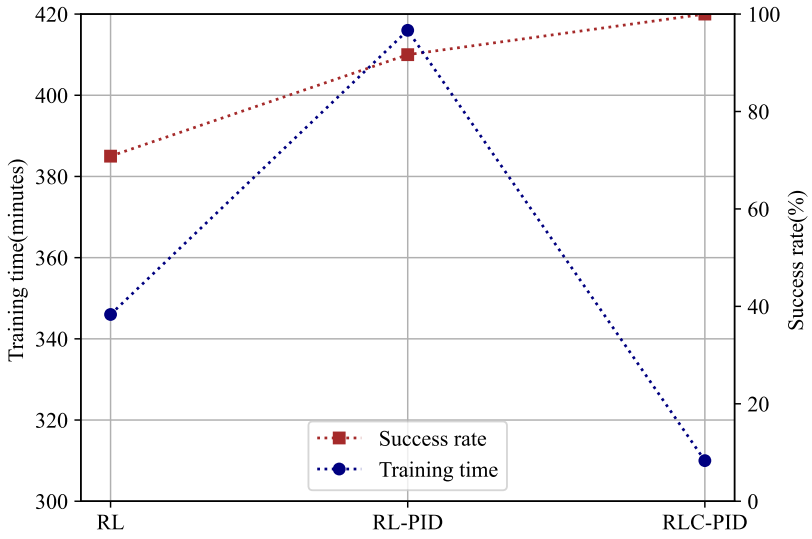


Figure 5. Comparison of success times and total used time (in minutes) among RL, RL-PID and RLC-PID during training, $N_{train} = 400$, $p_0 = (0, 0, 4.0)$.

In order to demonstrate the stability and convergence of the proposed method, we compared the accumulated reward of RL, RL-PID and RLC-PID in Figure 6, and we also compared the loss of RL, RL-PID and RLC-PID in Figure 7. The RL approach had the lowest reward, and RL-PID was close to RLC-PID during the training. Finally, the loss of the three approaches was close to zero, indicating that the learned policies became stable, although without guarantee of the high quality of the policies.

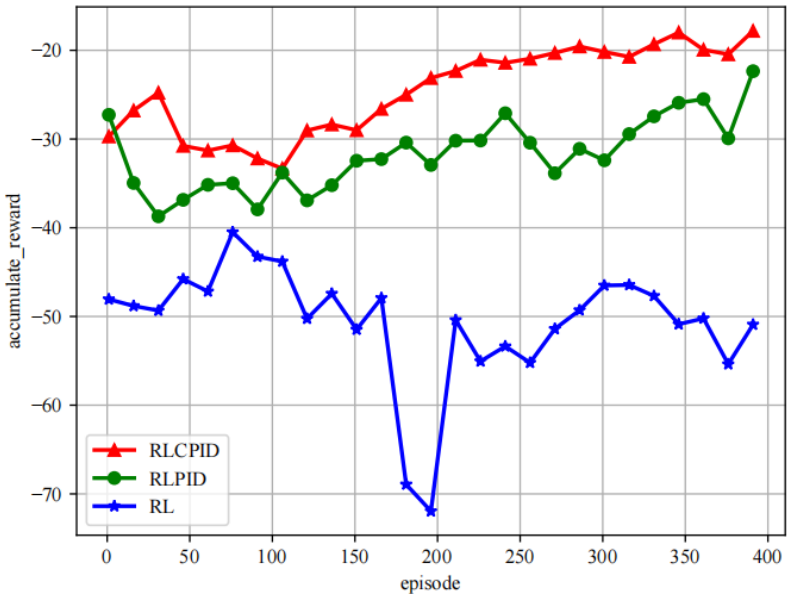


Figure 6. Comparison of the accumulated reward among RL, RL-PID, RLC-PID during training, $N_{train} = 400$, $p_0 = (0, 0, 4.0)$.

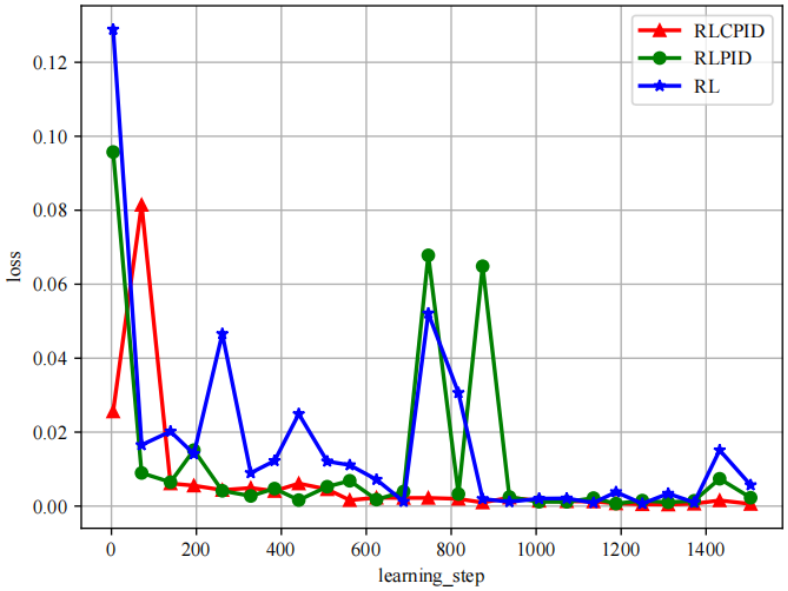


Figure 7. Comparison of loss among RL, RL-PID, RLC-PID during training, $N_{train} = 400$, $p_0 = (0, 0, 4.0)$.

4.3. Testing in the Simulation Environment

4.3.1. Testing with a Static Vehicle

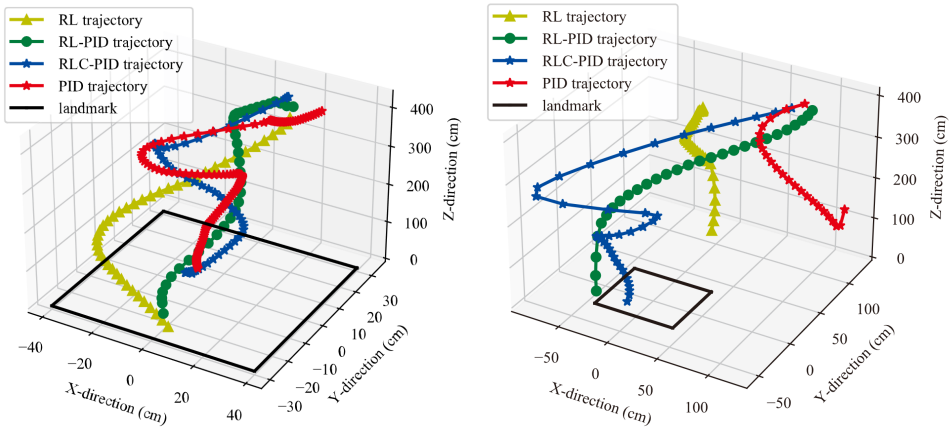
We tested the learned controllers together with the PID controller in a static scenario, with two initial landing conditions $p_0 = (0.2, 0.2, 4.0)$ and $p_0 = (1.0, 1.0, 4.0)$. The results of success times are compared in Table 1. It illustrated that the PID parameters of RLC-PID resulted in the best performance among the three approaches.

Table 1. Success times of landing on a static ground vehicle during testing, $N_{test} = 100$.

Approach	PID	RL	RL-PID	RLC-PID
$p_0 = (0.2, 0.2, 4.0)$	90	25	100	100
$p_0 = (1.0, 1.0, 4.0)$	52	0	55	92

The condition $p_0 = (0.2, 0.2, 4.0)$ was relatively easy because it was close to the condition $p_0 = (0, 0, 4.0)$ used for training. In other words, the marker was close to the center of field of view (FOV) of the UAV and easily tracked by the UAV. Accordingly, PID, RL-PID, RLC-PID solved it with high success rates, except that RL alone failed many times. In contrast, the condition $p_0 = (1.0, 1.0, 4.0)$ was more difficult as the marker was close to the boundary of the UAV's FOV. In other words, the UAV would lose track of the marker if it flew in the wrong direction. As a result, the performance of PID and RL-PID dropped almost by half while RLC-PID still maintained high performance.

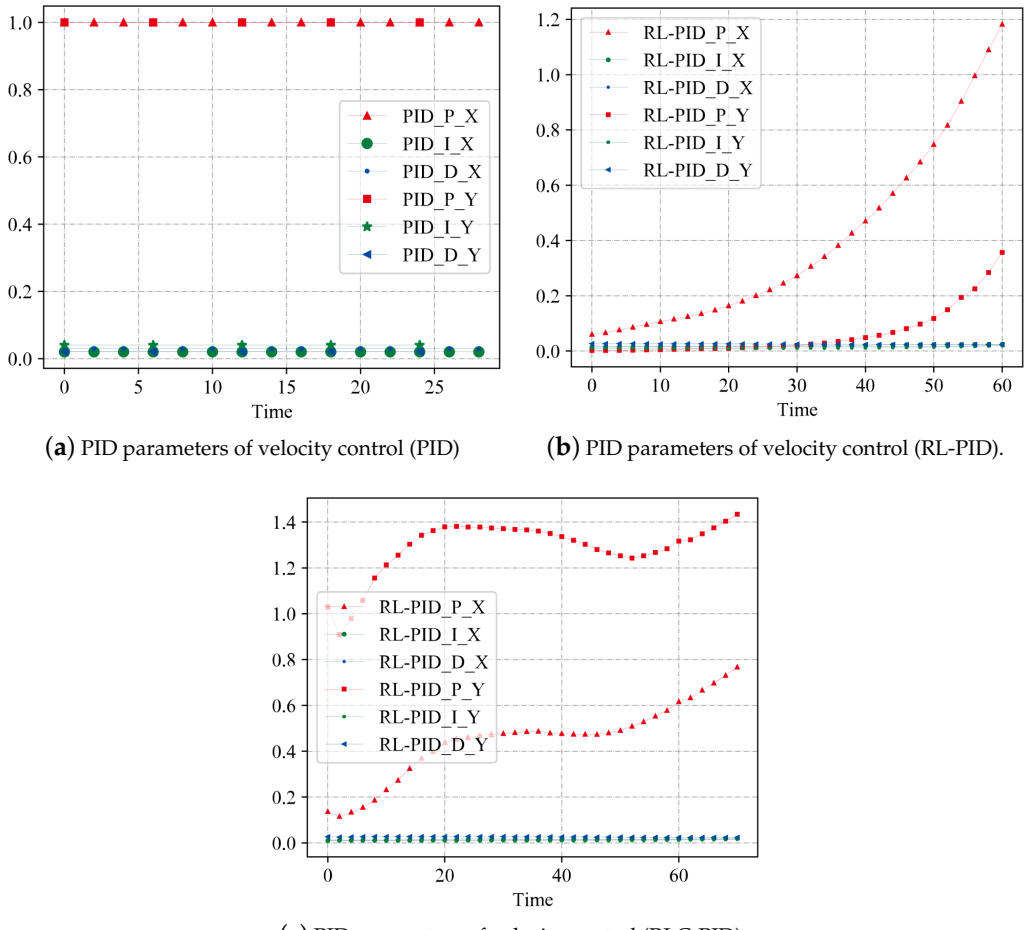
The trajectories of PID, RL, RL-PID and RLC-PID were also compared in Figure 8. We note that the RLC-PID approach encouraged a circular landing pattern compared with other approaches that showed longer trajectories of a vertical landing pattern. In other words, RLC-PID suggested the UAV speed up in the x and y directions in the beginning when the UAV was far from the goal location, and it suggested the UAV slow down in the end when it was close to the destination.



(a) From $p_0 = (0.2, 0.2, 4.0)$. **(b)** From $p_0 = (1.0, 1.0, 4.0)$.

Figure 8. Trajectories of landing on a static vehicle (PID, RL, RL-PID, RLC-PID).

The PID parameters of $(k_P^{v_x}, k_I^{v_x}, k_D^{v_x}, k_P^{v_y}, k_I^{v_y}, k_D^{v_y})$ were compared for PID, RL-PID and RLC-PID (see Figure 9). The P-gains of $k_P^{v_x}$ and $k_P^{v_y}$ showed different behaviors while other parameters almost remained unchanged.



(a) PID parameters of velocity control (PID) **(b)** PID parameters of velocity control (RL-PID).

(c) PID parameters of velocity control (RLC-PID)

Figure 9. PID parameters when landing on a static ground vehicle during testing.

Both $k_P^{v_x}$ and $k_P^{v_y}$ of PID remained around 1.0 according to human experience, while $k_P^{v_x}$ and $k_P^{v_y}$ of RL-PID kept increasing monotonically driven by the RL module. Due to

the use of corrective feedback, $k_P^{v_x}$ and $k_P^{v_y}$ of RLC-PID increased rapidly at the beginning and became slower afterward. On the other hand, the parameters of $k_I^{v_x}$, $k_I^{v_y}$, $k_D^{v_x}$ and $k_D^{v_y}$ remained close to zero for all three approaches.

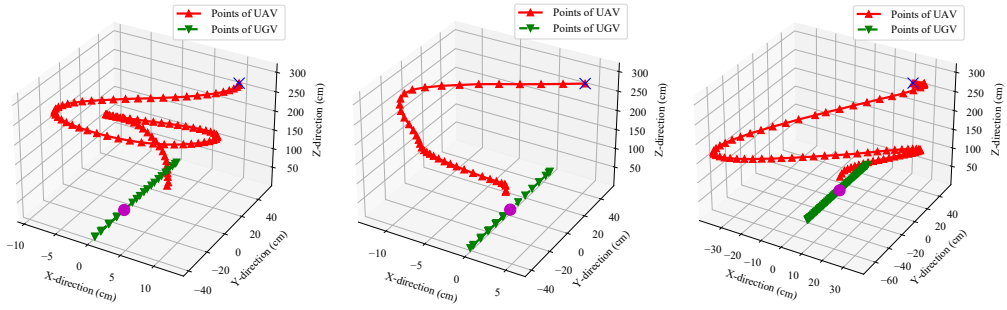
4.3.2. Testing with a Moving Vehicle

We tested 100 episodes for UAV landing on the moving ground vehicle from $p_0 = (0, 0, 4.0)$ using PID, RL-PID and RLC-PID methods, in which the vertical speed of UAV was set to $v_z = 0.1$ m/s and $v_z = 0.2$ m/s, respectively. The moving velocity of the ground vehicle was set to 0.1 m/s, but this information was unknown to the UAV, and the ground vehicle could occasionally move backward during the experiment. The PID parameters were set empirically as in the previous section. The task settings were the same with the static landing task, except that the ground vehicle was allowed to move back and forth in a straight line. Thus, it was more difficult for the UAV to land on the moving vehicle because the environment was changing with uncertainty. The results of success times were compared in Table 2, in which we can find the success times of each approach that was tested 100 times. It illustrates that the PID parameters of RLC-PID resulted in the best performance among the three approaches. Either the PID or the RL approach alone could hardly solve the landing task. For the RL-PID approach, the vertical speed of the UAV had a great influence on the success rate.

Table 2. Success times of landing on a moving ground vehicle during testing, $N_{test} = 100$.

Approach	PID	RL	RL-PID	RLC-PID
$v_z = 0.1$ m/s	10	0	91	97
$v_z = 0.2$ m/s	5	0	52	93

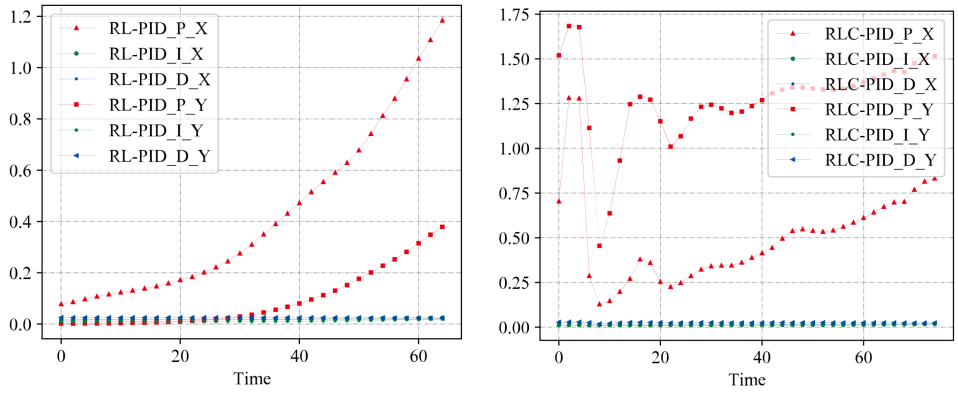
Figure 10 illustrates the trajectories of the UAV and the UGV in the experiments, where the blue cross represents the initial position of the UAV, the red line indicates the landing trajectory of the UAV, the green line represents the moving trajectory of the ground vehicle, and the purple dot represents the final position of the ground vehicle. An intuitive finding is that the UAV's landing trajectory of our approach is more smooth, and it can follow the motion of the ground vehicle.



(**a**) A failure (PID). (**b**) A success (RL-PID). (**c**) A success (RLC-PID).

Figure 10. Trajectories of landing on a moving vehicle (PID, RL-PID, RLC-PID), $p_0 = (0, 0, 4.0)$.

Figure 11 shows how the PID parameters of $(k_P^{v_x}, k_I^{v_x}, k_D^{v_x}, k_P^{v_y}, k_I^{v_y}, k_D^{v_y})$ were adapted for RL-PID and RLC-PID during the experiment. Similar to the testing results in the static vehicle setting, $k_P^{v_x}$ and $k_P^{v_y}$ of RL-PID kept increasing monotonically throughout the experiment, while $k_P^{v_x}$ and $k_P^{v_y}$ of RLC-PID changed more rapidly at the beginning, and changed slower afterward. As a result, the PID parameters had a significant influence on the landing trajectories and the success rates of the landing task.

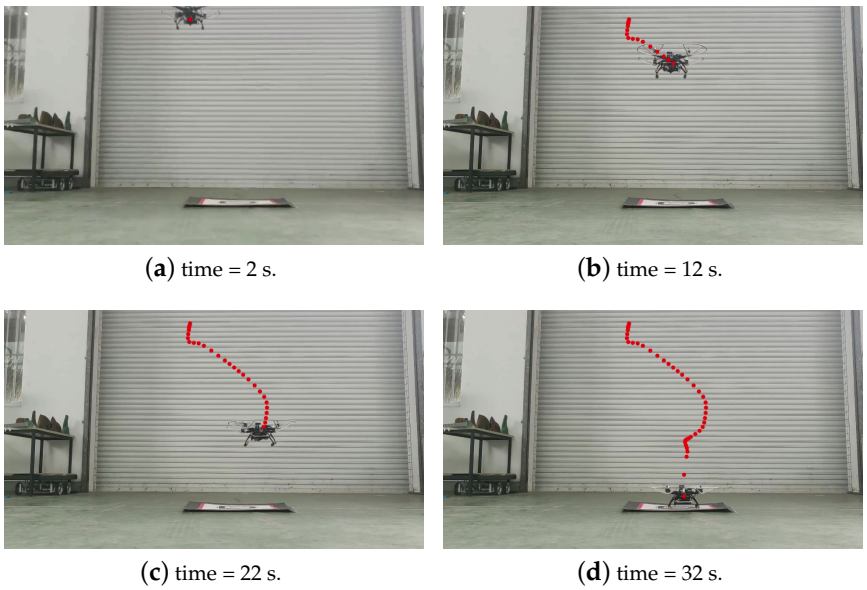


(**a**) PID parameters of velocity control (RL-PID). (**b**) PID parameters of velocity control (RLC-PID)

Figure 11. PID parameters when landing on a moving ground vehicle during testing.

4.4. Real-World Experiments

The models and parameters of the RLC-PID approach trained in the simulation were directly transferred to real-world experiments without any modification. The settings of real-world experiments were similar to the simulation. Due to safety concerns, we first tested the learned RLC-PID controller with a static landmark five times (see Figure 12). Here, the landmark was exactly the same as in the simulation environment. In all five tests, the UAV successfully landed on the landmark, and the final landing locations were all close to the center of the landmark.

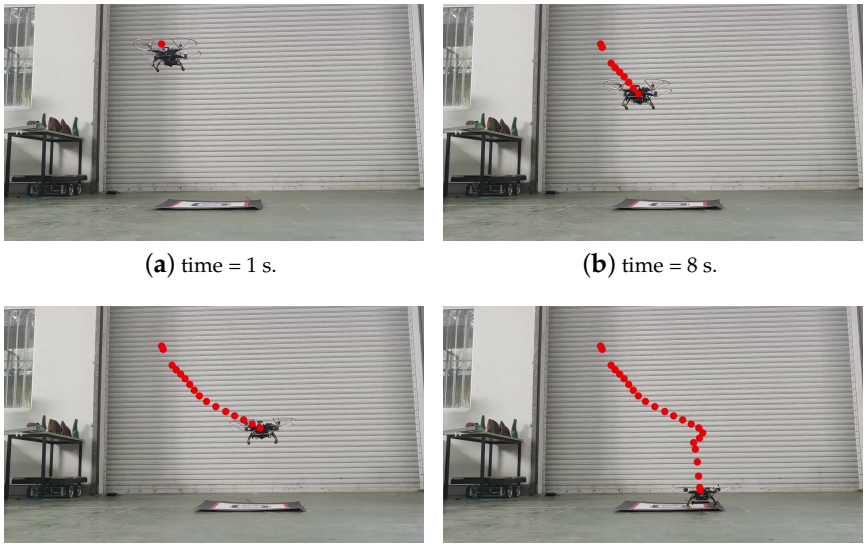


(**a**) time = 2 s. (**b**) time = 12 s.

(**c**) time = 22 s. (**d**) time = 32 s.

Figure 12. Real-world UAV landing on a static landmark.

Then, we used a movable landmark pulled by two human operators through two strings to imitate the case of landing on a mobile platform (see Figure 13). We note that the UAV had no information about the moving directions of the landmark. During the five tests, the landmark was pulled back and forth in random directions, and the results show that the UAV also successfully landed on the landmark, but the final landing positions had a larger deviation from the center position of the landmark compared with the static cases.

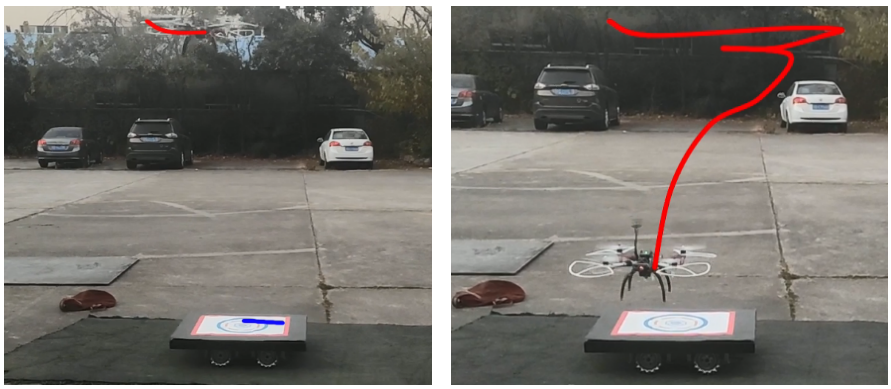


(**a**) time = 1 s. (**b**) time = 8 s.

(**c**) time = 15 s. (**d**) time = 22 s.

Figure 13. Real-world UAV landing on a movable landmark.

Finally, we tested the landing performance on a real mobile vehicle (see Figure 14). In the experiment, the moving velocity of the UGV was set to 0.1 m/s, but its moving direction was uncertain. We can also see from the figure that since the UAV needed to track and minimize the distance to the center of the landmark, its trajectory reflected the movement of the ground vehicle to some extent. In all five tests, we found that the UAV still managed to land on the ground vehicle successfully, but the final positions were close to the boundary of the landmark. In comparison with the indoor experiments, the outdoor experiments were affected by wind and airflow. Therefore, the landing accuracy was slightly worse than the indoor experiments.



(**a**) Early stage of landing. (**b**) Late stage of landing.

Figure 14. Real-world UAV landing on a mobile ground vehicle.

5. Conclusions and Future Work

In this paper, we have proposed an autonomous UAV landing approach by combining the advantages of the traditional PID control method and reinforcement learning. Specifically, we have designed an RL-PID framework that allows the RL module to adaptively adjust the parameters of the PID module in an online fashion. In addition, we have used corrective human feedback to provide immediate rewards to speed up the learning process. In both simulation and real-world experiments, we have demonstrated the effectiveness of the proposed RLC-PID algorithm in terms of success rate. The models and parameters of the RLC-PID controller trained in simulation could be directly transferred to real-world experiments without much fine-tuning. In future work, we will incorporate online human

intervention into our framework, and develop a more sophisticated credit assignment mechanism.

Author Contributions: Conceptualization, L.W.; data curation, P.Z.; formal analysis, C.W. (Chang Wang); funding acquisition, C.W. (Chang Wang); methodology, P.Z.; software, P.Z.; supervision, C.W. (Changyun Wei); writing—original draft, C.W. (Changyun Wei); writing—review and editing, C.W. (Chang Wang). All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Science and Technology Innovation 2030-Key Project of "New Generation Artificial Intelligence" under Grant 2020AAA0108200 and in part by the National Natural Science Foundation of China under Grant 61906203.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Liu, P.; Chen, A.Y.; Huang, Y.N.; Kang, S.C. A review of rotorcraft unmanned aerial vehicle (UAV) developments and applications in civil engineering. *Smart Struct. Syst.* **2014**, *13*, 1065–1094. [CrossRef]
2. Tsouros, D.C.; Bibi, S.; Sarigiannidis, P.G. A review on uav-based applications for precision agriculture. *Information* **2019**, *10*, 349. [CrossRef]
3. Ren, H.; Zhao, Y.; Xiao, W.; Hu, Z. A review of uav monitoring in mining areas: Current status and future perspectives. *Int. J. Coal Sci. Technol.* **2019**, *6*, 20–333. [CrossRef]
4. Michael, N.; Shen, S.; Mohta, K.; Kumar, V.; Nagatani, K.; Okada, Y.; Kiribayashi, S.; Otake, K.; Yoshida, K.; Ohno, K.; et al. Collaborative mapping of an earthquake damaged building via ground and aerial robots. *J. Field Robot.* **2012**, *29*, 832–841. [CrossRef]
5. Baca, T.; Stepan, P.; Spurny, V.; Hert, D.; Penicka, R.; Saska, M.; Thomas, J.; Loianno, G.; Kumar, V. Autonomous landing on a moving vehicle with an unmanned aerial vehicle. *J. Field Robot.* **2019**, *36*, 874–891. [CrossRef]
6. Talha, M.; Asghar, F.; Rohan, A.; Rabah, M.; Kim, S.H. Fuzzy logic-based robust and autonomous safe landing for uav quadcopter. *Arab. J. Sci. Eng.* **2019**, *44*, 2627–2639. [CrossRef]
7. 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]
8. Erginer, B.; Altug, E. Modeling and pd control of a quadrotor vtol vehicle. In Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istambul, Turkey, 13–15 June 2007; IEEE: New York, NY, USA, 2007; pp. 894–899.
9. Asadi, K.; Suresh, A.K.; Ender, A.; Gotad, S.; Maniyar, S.; Anand, S.; Noghabaei, M.; Han, K.; Lobaton, E.; Wu, T. An integrated ugv-uav system for construction site data collection. *Autom. Constr.* **2020**, *112*, 103068. [CrossRef]
10. Bacheti, V.; Brandao, A.; Sarcinelli-Filho, M. Path-following with a ugv-uav formation considering that the uav lands on the ugv. In Proceedings of the 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 1–4 September 2020; IEEE: New York, NY, USA, 2020; pp. 488–497.
11. Shaker, M.; Smith, M.N.; Yue, S.; Duckett, T. Vision-based landing of a simulated unmanned aerial vehicle with fast reinforcement learning. In Proceedings of the 2010 International Conference on Emerging Security Technologies, Canterbury, UK, 6–7 September 2010; IEEE: New York, NY, USA, 2010; pp. 183–188.
12. 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]
13. Lee, S.; Shim, T.; Kim, S.; Park, J.; Hong, K.; Bang, H. Vision-based autonomous landing of a multi-copter unmanned aerial vehicle using reinforcement learning. In Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS), Dallas, TX, USA, 12–15 June 2018; IEEE: New York, NY, USA, 2018; pp. 108–114.
14. Rodriguez-Ramos, A.; Sampedro, C.; Bavle, H.; Moreno, I.G.; Campoy, P. A deep reinforcement learning technique for vision-based autonomous multirotor landing on a moving platform. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; IEEE: New York, NY, USA, 2018; pp. 1010–1017.
15. Wang, S.; Yin, X.; Li, P.; Zhang, M.; Wang, X. Trajectory tracking control for mobile robots using reinforcement learning and PID. *Iran. J. Sci. Technol. Trans. Electrical Eng.* **2020**, *44*, 1059–1068. [CrossRef]
16. Carlucho, I.; De Paula, M.; Villar, S.A.; Acosta, G.G. Incremental q-learning strategy for adaptive pid control of mobile robots. *Expert Syst. Appl.* **2017**, *80*, 183–199. [CrossRef]
17. Watkins, C.J.C.H.; Dayan, P. Technical note: Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
18. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic policy gradient algorithms. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; PMLR: New York, NY, USA, 2014; pp. 387–395.

19. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347
20. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the 2018 International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; PMLR: New York, NY, USA, 2018; pp. 1861–1870.
21. Ross, S.; Gordon, G.; Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Lauderdale, FL, USA, 11–13 April 2011; pp. 627–635.
22. Kelly, M.; Sidrane, C.; Driggs-Campbell, K.; Kochenderfer, M.J. Hg-dagger: Interactive imitation learning with human experts. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; IEEE: New York, NY, USA, 2019; pp. 8077–8083.
23. Spencer, J.; Choudhury, S.; Barnes, M.; Schmittle, M.; Chiang, M.; Ramadge, P.; Srinivasa, S. Learning from Interventions: Human-robot interaction as both explicit and implicit feedback. In *Robotics: Science and Systems*; MIT Press Journals: Cambridge, MA, USA, 2020. [CrossRef]
24. Knox, W.B.; Stone, P. Interactively shaping agents via human reinforcement: The TAMER framework. In Proceedings of the Fifth International Conference on Knowledge Capture, Redondo Beach, CA, USA, 1–4 September 2009; pp. 9–16.
25. Celemin, C.; Ruiz-del-Solar, J. An interactive framework for learning continuous actions policies based on corrective feedback. *J.Intell. Robot. Syst.* **2019**, *95*, 77–97. [CrossRef]
26. Celemin, C.; Maeda, G.; Ruiz-del-Solar, J.; Peters, J.; Kober, J. Reinforcement learning of motor skills using policy search and human corrective advice. *Int. J. Robot. Res.* **2019**, *38*, 1560–1580. [CrossRef]
27. Scholten, J.; Wout, D.; Celemin, C.; Kober, J. Deep reinforcement learning with feedback-based exploration. In Proceedings of the IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019; IEEE: New York, NY, USA, 2019; pp. 803–808.
28. Zhang, H.T.; Hu, B.B.; Xu, Z.; Cai, Z.; Liu, B.; Wang, X.; Geng, T.; Zhong, S.; Zhao, J. Visual navigation and landing control of an unmanned aerial vehicle on a moving autonomous surface vehicle via adaptive learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 5345–5355. [CrossRef] [PubMed]
29. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjel, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [PubMed]
30. VanHasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30, pp. 2094–2100.
31. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Santa Monica, CA, USA, 28 September–2 October 2004; IEEE: New York, NY, USA, 2004; pp. 2149–2154.
32. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An Open-Source Robot Operating System. In Proceedings of the 2009 ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009.