Automatic Control Optimization for Large-Load Plant-Protection Quadrotor

Dada Hu, Zhongcai Pei and Zhiyong Tang *

School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China; by1603163@buaa.edu.cn (D.H.); peizc@buaa.edu.cn (Z.P.) * Correspondence: zyt_76@buaa.edu.cn; Tel.: +86-10-82339730

Abstract: In this article, methods for the attitude control optimization of large-load plant-protection quadrotor unmanned aerial vehicles (UAVs) are presented. Large-load plant-protection quadrotors can be defined as quadrotors equipped with sprayers and a tank containing a large amount of water or pesticide, allowing the quadrotors to water plants or spray pesticide during flight. Compared to the control of common small quadrotors, two main points need to be considered in the control of large-load plant-protection quadrotors—first, the water in the tank gradually diminishes during flight and the physical parameters change during this process. Second, the size and mass of the rotors are especially large, which greatly slows the response rate of the rotors. We present an extended-state reinforcement learning (RL) algorithm to solve these problems. The moment of inertia (MOI) of the three axes and the dynamic response constant of the rotors are included in the state list of the quadrotor during the training process, so that the controller can learn these changes in the models. The controlling laws are automatically generated and optimized, which greatly simplifies the tuning process compared to those of traditional control algorithms. The controller in this article is tested on a 10 kg class large-load plant-protection quadrotor, and the flight performance verifies the effectiveness of our work.

Keywords: quadrotor attitude control; large-load plant-protection quadrotor; reinforcement learning; deterministic policy gradient

1. Introduction

Quadrotor unmanned aerial vehicles (UAVs) have been widely used in many areas because of their maneuverability and flexibility, as Reference [1] summarized. In plant protection areas, quadrotor UAVs are helping reducing agriculture damage [2]. However, quadrotors are high-dimensional nonlinear underactuated systems with different physical parameters, which increases the difficulty of stable control. Some control algorithms have been tested and applied to quadrotors, and the most widely used algorithm is the traditional proportional-integral-derivative (PID) controller, which achieves an acceptable flight performance. Projects [3,4] are popular open source flight controllers using PID-based control algorithms; the users must tune PID parameters for each quadrotor platform. Another research effort [5] made many improvements to the PID controllers, and the Smith predictor and rotational trajectory planner were augmented. The linear quadratic regulator (LQR) algorithm has also made progress in the control of quadrotor UAVs. One study [6] linearized the quadrotor system and applied the LQR method to quadrotors and energy consumption was also considered. Another report [7] presented a morphing quadrotor, where the arms of the quadrotor could rotate around the body using servos. To achieve stable flight at any configuration, the researchers exploited an LQR-based control strategy that adapts to the drone morphology in real time. However, the performance of these linear control algorithms becomes worse with a larger attitude because the model errors are larger under this condition. Studies [5,7] addressed this problem and compensated for the model errors in their attitude control loops.
Many studies have tested nonlinear control algorithms on quadrotor UAVs. One study [8] achieved stable control of a quadrotor with the backstepping algorithm, and this control method was found to control the quadrotor in any state without nonlinear compensation. Backstepping and sliding mode techniques were applied to an indoor micro quadrotor in another study [9]. A further study [10] developed sliding mode controllers for the attitude loop and the position loop separately. H-infinity control was found to achieve robust control for quadrotors in another study [11], and the effect of external disturbances was minimized. An additional study [12] introduced observers to further compensate for these external disturbances.

The present article focuses on the attitude control of large-load plant-protection quadrotors. Large-load plant-protection quadrotors can be defined as quadrotors equipped with sprayers and a tank containing a large amount of water or pesticide (the mass of water or pesticide can be even heavier than the quadrotors themselves) and the quadrotors can water plants or spray pesticide during flight. Compared to the control of common small quadrotors, some factors need to be considered in the control of large-load plant-protection quadrotors. First, the water in the tank gradually diminishes during application, so the physical parameters change during this process. The mass and moment of inertia (MOI) of the three axes of a plant-protection quadrotor with an empty tank or with a full tank can be very different. The controller must learn these changes because the flight performance of the plant-protection quadrotor can be severely affected. One report [13] presents a fuzzy PID method to solve this problem on plant-protection quadrotor, the PID parameters were adjusted in real time using a fuzzy control table. This problem also appears in the control of the morphing quadrotor in the study [7], and was solved using the adaptive LQR method.

Deterministic artificial intelligence (DAI) is a novel intelligent control algorithm that can automatically enforce optimality without any controller tuning. This method has been applied to unmanned underwater vehicles (UUV) [14] and spacecraft [15], and is also able to solve the changing-MOI problem. Second, for large-load plant-protection quadrotors, the size and mass of the rotors are particularly large, which greatly slows the rotor response rate. This factor cannot be ignored; if the response rate is too low, the quadrotor cannot track the desired attitude in a timely manner in any case, which limits the size of electrically powered rotors. Projects [3,4] present feedforward compensation methods to solve this problem. In [5], a Smith predictor is designed to accelerate the response rate of rotors. Another report [7] includes the delay of the rotors in the state equation of the system and solves for the optimal control law using the LQR method. These methods can achieve acceptable performance but rely heavily on parameter tuning, which can be burdensome and dangerous in real applications.

We present a reinforcement learning (RL) method to solve the above problems. RL is an optimization algorithm, where the agents take actions and receive rewards in the environment. In the training process, the agents will optimize the policy to get higher rewards. RL has made remarkable achievements and demonstrated strong potential in complex nonlinear control systems. These techniques have been tested on quadrotor UAVs. One study [16] controlled the full quadrotor using only one neural network in an end-to-end style; the controller tracked the desired position and sent control values to the four rotors directly. In [17], a supplementary RL controller was designed to optimize the performance of a PID-based controller. Another article [18] studied the fault-tolerant control (FTC) problem using the RL method, and a simulation example of a quadrotor was conducted. The report [19] proposes an event-triggered RL method to control the quadrotor, and achieved great performance. However, these RL methods focus on specific platforms and cannot be applied to platforms with other physical parameters or adapt to changes in the physical parameters of the platform. We presented an extended-state RL method in our previous research [20], where the core idea is to extend the critical physical parameters of the system into the state list during the RL training process, and then the controller can automatically fit changes in these parameters. In the present article, we choose the MOIs of three axes and the dynamic response constant of the rotors as critical physical
parameters. The RL method in this article is a deterministic policy gradient (DPG) [21] style algorithm. To accelerate the training process, we train 128 quadrotors with different physical parameters in parallel.

Our method automatically generates and optimizes the control law with almost no parameter tuning, which is a great benefit compared to many traditional algorithms. This end-to-end controller design process was found to simplify tuning work and to ensure safety during the experimental process, which is very important for large-load quadrotors. The controller is run on flight controller hardware developed by our team named TensorFly Project that is equipped with a high-performance Raspberry Pi microcomputer, which meets the computational cost of our controller neural network.

The article is organized as follows—in Section 2.1, a nonlinear model of the large-load plant-protection quadrotor system model is established and identification methods are introduced. In Section 2.2, the training process of the extended-state RL algorithm are presented. In Sections 3.1–3.4, we introduce the training details and results. Sections 3.5 and 3.6 present the real flight performance of a 10 kg class large-load plant-protection quadrotor using the RL controller. Section 4 concludes the article.

2. Materials and Methods

2.1. System Model and Identification

In this article, the quadrotor is a 10 kg class large-load plant-protection quadrotor, and Figure 1 introduces the definition of the quadrotor’s frames. In Figure 1a, the attitude of the quadrotor is represented with $\phi$ (the roll angle), $\varphi$ (the pitch angle) and $\psi$ (the yaw angle). In Figure 1b, $\omega_1$ to $\omega_4$ represent the angular velocities of the four rotors.

![Coordinate definition of the quadrotor.](image)

Figure 1. Coordinate definition of the quadrotor.

The movement and the rotation of the quadrotor is determined by the lift force of the rotors. The dynamic model of the quadrotor is presented in Equation (1):

\[
\begin{align*}
J_{xx} \dot{\omega}_x &= \omega_y \omega_z (J_{yy} - J_{zz}) + M_x \\
J_{yy} \dot{\omega}_y &= \omega_x \omega_z (J_{zz} - J_{xx}) + M_y \\
J_{zz} \dot{\omega}_z &= \omega_x \omega_y (J_{xx} - J_{yy}) + M_z,
\end{align*}
\]

where $\omega_x, \omega_y, \omega_z$ represent the angular velocities of the three axes and $J_{xx}, J_{yy}, J_{zz}$ represent the MOIs of the three axes. $M_x, M_y, M_z$ represent the torque generated by the rotors. The torque is generated by the four rotors:

\[
\begin{align*}
M_x &= \frac{\sqrt{2}}{2} C_T \, d \, C_x \\
M_y &= \frac{\sqrt{2}}{2} C_T \, d \, C_y \\
M_z &= C_M \, C_z
\end{align*}
\]
\begin{align}
C_x &= -\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2 = S_x a_x \\
C_y &= -\omega_1^2 + \omega_2^2 + \omega_3^2 - \omega_4^2 = S_y a_y \\
C_z &= -\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2 = S_z a_z,
\end{align}

where $C_T$ and $C_M$ are the lift coefficient and the torque coefficient of the rotors, and $\omega_1$ to $\omega_4$ are rotors’ angular velocities. In addition, $d$ represents the arm length of the plant-protection quadrotor; $C_x, C_y, C_z$ and $S_x, S_y, S_z$ are intermediate variables; and $a_x, a_y, a_z \in [-1.0, 1.0]$ are the control values of the attitude controller. We scale the controller output between $-1.0$ and $1.0$ to simplify the design of controller neural networks. The relationships between the angular velocity of a rotor and the control value of the rotor are modeled in Equation (4):

\begin{equation}
\begin{aligned}
\omega &= \frac{1}{T_m+1}\omega_d \\
\omega_d &= C_R \sigma,
\end{aligned}
\end{equation}

where $\omega$ represents the current angular velocity of the rotor and $\omega_d$ represents the desired angular velocity of the rotor. $\sigma \in [0, 1.0]$ represents the control value of the rotor, $C_R$ represents the rotor control constant, and $T_m$ represents the dynamic response constant.

System identification is an important aspect in controlling quadrotors and we present several methods to measure or estimate the relevant physical parameters (shown in Figure 2). Parameters such as the mass and length of arms can be measured directly. MOIs can be measured using the three wire pendulum methods, but these methods require large and expensive equipment for a large-load quadrotor. We estimate the MOIs using the computer aided design (CAD) method; that is, we build an accurate model in SolidWorks software (Figure 2a), weigh every component of the quadrotor, assign the masses of these components in SolidWorks software, and finally automatically calculate the MOIs of the total quadrotor. Moreover, this article focuses on controlling the plant-protection quadrotor with any amount of water carried on it, and the water model must be considered. It is worth noting that, the water in the tank may wobble during flight, which can greatly worsen the flight performance. To increase the impact of water, we control the quadrotor within a low speed (under 5.0 m/s in the $x$ and $y$ axes, under 1.0 m/s in the $z$ axis) and a low accelerated velocity (under 1.0 m/s$^2$ in the three axes). Moreover, the tank in this project is an anti-oscillation tank specially designed for plant-protection quadrotor, where water is separated in small blocks, which greatly eliminate the impact of water. As a result, we assume that the water is shaped as a cuboid, as shown in Figure 3.

In Figure 3, $\text{cog}_{bo}$ is the center of gravity (COG) of the body (including the empty tank), $\text{cog}_{wa}$ is the COG of the water cube, and $\text{cog}_{all}$ is the COG of the total quadrotor. In addition, $d_{\text{cog}}$ denotes the distance between $\text{cog}_{bo}$ and $\text{cog}_{wa}$, $d_{bo}$ represents the distance between $\text{cog}_{bo}$ and $\text{cog}_{all}$, and $d_{wa}$ is the distance between $\text{cog}_{wa}$ and $\text{cog}_{all}$. To simplify the calculation, the movement of the COG of the water is ignored, so $d_{\text{cog}}$ is constant. $V_{wa}$ is the volume of the water cube, and $l_{wa}$ is the length and width of the water cube, so the height...
of the water cube can be calculated as 

\[ h_{wa} = \frac{V_{wa}}{l_{wa}^2} \]

Then, \( d_{bo} \) and \( d_{wa} \) can be calculated as Equation (5) presents:

\[
\begin{align*}
  d_{bo} &= \frac{V_{wa} l_{wa}^2}{m_{bo} + m_{wa} d_{cog}} d_{cog} \\
  d_{wa} &= \frac{m_{bo}}{m_{bo} + m_{wa} d_{cog}} d_{cog},
\end{align*}
\]

(5)

where \( \rho_{wa} \) is the density of the water or pesticide. The MOIs of the water can be calculated using Equation (6):

\[
\begin{align*}
  J_{xx,wa} &= \frac{V_{wa} \rho_{wa}}{12} (l_{wa}^2 + h_{wa}^2) \\
  J_{yy,wa} &= \frac{V_{wa} \rho_{wa}}{12} (l_{wa}^2 + h_{wa}^2) \\
  J_{zz,wa} &= \frac{V_{wa} \rho_{wa} l_{wa}^2}{6},
\end{align*}
\]

(6)

where \( J_{xx,wa} \), \( J_{yy,wa} \), and \( J_{zz,wa} \) are the MOIs of the water cube in the three axes. Then, the MOIs of the total quadrotor can be calculated using rigid-body parallel axis theory, as Equation (7) presents:

\[
\begin{align*}
  J_{xx} &= J_{xx,bo} + J_{xx,wa} + V_{wa} \rho_{wa} d_{wa}^2 + m_{bo} d_{bo}^2 \\
  J_{yy} &= J_{yy,bo} + J_{yy,wa} + V_{wa} \rho_{wa} d_{cog}^2 + m_{bo} d_{bo}^2 \\
  J_{zz} &= J_{zz,bo} + J_{zz,wa},
\end{align*}
\]

(7)

where \( J_{xx,bo} \), \( J_{yy,bo} \), and \( J_{zz,bo} \) are the MOIs of the quadrotor body in the three axes. In this way, we can estimate the MOIs of the total quadrotor with any amount of water carried in it.

Figure 3. Model of the large-load quadrotor with water in the tank.

In this article, we identify the parameters of the motors and rotors with a rotor bench (Figure 2b). The bench can measure the lift force, the torque and the angular velocity of the rotor, so the lift coefficient \( C_T \) and torque coefficient \( C_M \) can be calculated. The rest of the unknown parameters, such as the dynamic response constant of the rotors \( T_m \), can be found in an online quadrotor database: https://flyeval.com/ (accessed on 27 March 2021). [22].

In the simulation and training process, the simulation environment updates the states of the quadrotors, including the attitude, the angular velocities of the three axes and the angular velocities of the rotors. The attitude is described in unit quaternions in this article:

\[
\begin{align*}
  q_0 &= \cos \left( \frac{\Theta}{2} \right) \\
  q_1 &= \sin \left( \frac{\Theta}{2} \right) p_x \\
  q_2 &= \sin \left( \frac{\Theta}{2} \right) p_y \\
  q_3 &= \sin \left( \frac{\Theta}{2} \right) p_z,
\end{align*}
\]

(8)
where $\Theta$ is the combined rotation of the roll angle, the pitch angle and the yaw angle, and $p_x, p_y, p_z$ are the projections of $\Theta$ in the roll, pitch and yaw axes, respectively. The quaternions can be updated with angular velocities in three axes using the Runge-Kutta method:

$$
\begin{bmatrix}
q_0 \\
q_1 \\
q_2 \\
q_3
\end{bmatrix} = \frac{1}{2} \begin{bmatrix}
0 & -\omega_x & -\omega_y & -\omega_z \\
\omega_z & 0 & -\omega_x & -\omega_y \\
\omega_y & -\omega_z & 0 & -\omega_x \\
\omega_x & \omega_y & \omega_z & 0
\end{bmatrix} \begin{bmatrix}
q_0 \\
q_1 \\
q_2 \\
q_3
\end{bmatrix}
$$

(9)

2.2. Attitude Controller Design for Large-Load Plant-Protection Quadrotors

The attitude controller receives the desired attitude $att_d$, the current attitude $att$, the angular velocities in the three axes $\omega_x, \omega_y, \omega_z$, the MOIs in the three axes $J_{xx}, J_{yy}, J_{zz}$ and the response constant of the rotors $T_m$, and then outputs the control values of the three axes $a_x, a_y, a_z \in [-1.0, 1.0]$ according to the states. The control allocator transfers $a_x, a_y, a_z$ and the desired throttle value $th_d$ to the control values of each rotor $\sigma_i, i = 1, 2, 3, 4$. The control diagram is presented in Figure 4.

Figure 4. Control diagram for large-load plant-protection quadrotors.

2.2.1. Quadrotor Attitude Controller with Reinforcement Learning

The RL frame includes the following factors—the agent, environment, state $s_t$, action $a_t$, policy $\pi(s_t)$ and reward $r_t(s_{t+1}, a_t)$. During the training process, the environment outputs the state $s_t$ and the agent takes action $a_t$ with the policy $\pi(s_t)$ and then obtains a reward $r_t(s_{t+1}, a_t)$. To describe the performance of the agent in the long term, the value of the current state and action $V_t(s_t, a_t)$ is introduced:

$$V_t = r_t + \lambda r_{t+1} + \lambda^2 r_{t+1} + ... = r_t + \lambda V_{t+1},
$$

(10)

where $\lambda \in [0, 1)$ is the discount factor. Equation (10) shows that the stat-action value $V_t$ is the sum of expected future rewards. Although these rewards are unknown, we can estimate them from previous outcomes. We can assume that an optimal policy $\pi_{opt}$ can maximize $V_{t+1}$:

$$V_t(s_{t+1}, a_t) = r_t(s_{t+1}, a_t) + \lambda \max_{a_{t+1} \in A} (V_{t+1}(s_{t+1}, a_{t+1}))$$

$$= r_t(s_{t+1}, a_t) + \lambda V_{t+1}(s_{t+1}, \pi_{opt}(s_{t+1})),$$

(11)

where $A$ is the action space of the agent. During the training process, we apply the current policy $\pi$ instead of $\pi_{opt}$.

In this article, the agent is a quadrotor with four rotors, and the environment receives the angular velocities of the rotors and outputs the attitude and angular velocities in three axes.

2.2.2. Network Structure and Training Algorithm

The structure of the RL controller is an actor-critic [21] based structure, as presented in Figure 5. The critic network is built to estimate the value of the state and action, and the actor network is built to compute the control values with the state.
Figure 5. The RL networks structure.

The RL method in this article is the extended-state RL proposed in our previous work [20], where system parameters can be extended into the state list, which enables the controller to fit the changes in these parameters. As a result, the state list consists of the attitude errors \( att = [q_0, q_1, q_2, q_3] \) and the current angular velocities in the three axes \([\omega_x, \omega_y, \omega_z]\), along with the MOIs in the three axes \( J_{xx}, J_{yy}, J_{zz} \) and the response constant of the rotors \( T_m \). The action list consists of control values in the three axes \( a_x, a_y, a_z \).

The training method in this article is the DPG method [21], consists of two actor networks and two critic networks. The actor networks include the target actor network and the online actor network, both with the same structure, and two hidden layers of 128 nodes are included. The tanh activation function is set to limit the output value between \(-1.0 \) and \(1.0\) in the output layer. The online actor network is trained in each time step, the gradients are computed with the sampled policy gradients, and the weights are updated using the Adam optimizer [23]. The target actor network is soft updated [21] using the online actor network.

The critic networks also include the target critic network and the online critic network. The two critic networks share the same structure as well, and two hidden layers of 128 nodes are included. The squared error function is used in the critic network, and the cost function of the online critic network is a squared error function, and we lower the cost with the Adam optimizer. We soft update the weights of the target critic network using the online critic network.

The reward function design is critical work in the RL algorithm. Squared error style reward functions are widely used in the RL-based control domain, as Equation (12) presents:

\[
r(s) = -p_1 \cdot \Theta^2 - p_2 \cdot (\omega_x^2 + \omega_y^2 + \omega_z^2),
\]

where \(p_1\) and \(p_2\) are positive parameters and \(\Theta = 2arccos(q_0)\) is the total rotation angle of the attitude error. This reward function encourages the agent to stably reach the zero point. However, in many RL studies using these squared error reward functions, controllers show obvious steady-state errors [24]. We analyze the cause of steady-state errors and present a reward shaping method to solve this problem in article [20]. The reward shaping skill is also applied in this article, and we construct a piecewise function as:

\[
fr(x, \epsilon, \varrho) = \begin{cases} 
-|x|/\epsilon + 1, & (|x| \leq \epsilon) \\
(|x| - \epsilon)/(\epsilon - \varrho), & (\epsilon < |x| < \varrho) 
\end{cases}
\]

where \(x\) is the input state value (attitude error or angular velocity in this article), \(\epsilon\) and \(\varrho\) are constants. With Equation (13), the reward function is designed as:

\[
r(s) = p_1 \cdot fr(\Theta, \delta_1, \epsilon_1) + p_2 \cdot (fr(\omega_x, \delta_2, \rho_2) + fr(\omega_y, \delta_2, \rho_2) + fr(\omega_z, \delta_2, \rho_2)),
\]
where \( p_1, p_2, \delta_1, \rho_1, \delta_2 \) and \( \rho_2 \) are positive parameters to be tuned. Compared to the reward functions like Equation (12), this function output positive reward for close-to-zero states.

In a real flight environment, sensor noise can decrease the performance of the controller. The attitude angle data are estimated and filtered using gyroscope data, accelerometer data and magnetic compass data, so the errors of attitude angle data cannot be ignored. The angular velocities from sensors are relatively accurate, so errors of angular velocity data are ignored in this research. To improve the the robustness for sensor noise of the quadrotor, sensor noise are added to the agents in the training process.

In our algorithm, we run several agents in parallel and immediately employ their experience at each time step, which has some advantages:

1. Different agents have different physical parameters with different initial states and take different actions, thus avoiding the experience correlation problem.

2. Compared to traditional RL methods such as deep Q learning (DQN) [25], parallel training does not need experience replay skills and thus requires much less memory.

3. Several agents can provide more experience at the same time, greatly accelerating the training process.

The whole algorithm is listed as Algorithm 1.

**Algorithm 1** Extended-state RL parallel training algorithm

\( n_{agent} \): max number of agents

\( \epsilon_{max} \): max number of episodes

\( s_{max} \): max number of steps during an episode

\( R(s') \): The reward function

\( \tau \): soft update rate

1: create \( n_{agent} \) agents with different system parameters

2: randomly initialize the weights of the online critic network \( \theta_{v} \) and the online actor network \( \theta_{\pi} \)

3: initialize the weights of the target critic network \( \theta'_{v} \) and the target actor network \( \theta'_{\pi} \)

4: for \( epi = 1, 2...\epsilon_{max} \) do

5: initialize all agents with random states

6: for \( step = 1, 2...s_{max} \) do

7: clear the memory box \( B \leftarrow \emptyset \)

8: for \( n = 1, 2...n_{agent} \) do

9: receive the current state \( s \)

10: take action \( a \) according to \( s \)

11: receive the next state \( s' \)

12: put \( E = (s, a, s') \) into the memory box \( B \leftarrow B \cup E \)

13: end for

14: for \( i = 1, 2...n_{agent} \) do

15: obtain experience \( e_i = (s_i, a_i, s'_i) \) from \( B \)

16: compute the reward of experience \( e_i : r_i \leftarrow R(s'_i) \)

17: \( y_i \leftarrow r_i + \lambda V((s'_i, \pi(s'_i|\theta_{\pi})))|\theta_{v} \)

18: update \( \theta_{v} \) by minimizing the loss:

\[
L = \frac{1}{N} \sum y_i - V(s, a|\theta_{v})^2
\]

19: update \( \theta_{\pi} \) using the sampled policy gradients:

\[
\nabla_{\theta_{\pi}} \approx \frac{1}{N} \sum \nabla_{a} V((s, a|\theta_{v}))|s = s_i, a = a_i, \pi(s|\theta_{\pi}))|s = s_i
\]

20: soft update \( \theta'_{v} \) with \( \theta_{v} : \theta'_{v} \leftarrow \tau \theta_{v} + (1 - \tau) \theta'_{v} \)

21: soft update \( \theta'_{\pi} \) with \( \theta_{\pi} : \theta'_{\pi} \leftarrow \tau \theta_{\pi} + (1 - \tau) \theta'_{\pi} \)

22: end for

23: end for

24: end for
3. Results

3.1. Training Environment

A simulation environment is developed to train the controller using Python 3.8 and TensorFlow 1.14. We can create quadrotors with desired physical parameters and test the controllers in it. The time steps are not associated with real time, so the training process can be much faster than in the real world. The states of the quadrotors do not update in each time step until the training or control process is finished. We train the controller using a laptop computer equipped with an i7-7700HQ central processing unit (CPU) and an NVIDIA GTX1060 graphics processing unit (GPU).

3.2. Quadrotor Details

The large-load plant-protection quadrotor is a 10 kg class “X” type plant-protection quadrotor equipped with HLY Q9L brushless motors, 3080 carbon fiber propellers, Hobbywing XRotor Pro HV 80A electronic speed controls (ESCs) and two ACE TATTU 16,000 mAh 6S Li-polymer batteries in series. The parameters of the quadrotor with an empty tank are identified in Section 2.1 and listed in Table 1. The water in the tank is idealized as a cube with equal length and width.

Table 1. Physical parameters of the quadrotor with an empty tank.

| Parameter                    | Value          |
|------------------------------|----------------|
| Mass m                       | 8 kg           |
| Moment of inertia J_{xx}     | 1.066 kg·m²    |
| Moment of inertia J_{yy}     | 0.961 kg·m²    |
| Moment of inertia J_{zz}     | 1.843 kg·m²    |
| Arm length d                 | 0.65 m         |
| Lift coefficient C_T         | 5.7894 × 10⁻⁴ N/(rad/s)² |
| Torque coefficient C_M       | 2.283 × 10⁻⁵ N·m/(rad/s)² |
| Rotor control constant C_R   | 316 rad/s      |
| Rotor dynamic response constant T_m  | 0.035 s |
| Volume of the tank V_{tank, max} | 10 L          |
| Maximum mass of water M_{wa, max} | 10 kg         |
| Density of water ρ_{wa}      | 1.0 × 10³ kg/m³ |
| Distance between the quadrotor and water d_{cog} | 0.3 m          |
| Length and width of the water cube l_{wa} | 0.28 m         |

3.3. Training Details

The rotor dynamic response constant of this large-load quadrotor is T_m = 0.035 s, and we calculate that this constant for a common small quadrotor is T_m = 0.0157 s. During the training process, T_m is generated randomly between 0.015 s and 0.045 s for agents.

Using the data in Table 1, we can calculate the maximum and minimum MOIs in three axes using Equation (7), and the results are listed in Table 2:

Table 2. Range of MOIs in three axes.

| Parameter                     | Value          |
|-------------------------------|----------------|
| Maximum MOI in the x-axis     | 1.540 kg·m²    |
| Minimum MOI in the x-axis     | 1.066 kg·m²    |
| Maximum MOI in the y-axis     | 1.435 kg·m²    |
| Minimum MOI in the y-axis     | 0.961 kg·m²    |
| Maximum MOI in the z-axis     | 1.973 kg·m²    |
| Minimum MOI in the z-axis     | 1.843 kg·m²    |

As a result, the MOIs in the x and y axes range between 0.8 and 1.7 kg·m², and the MOI in the z axes ranges between 1.8 and 2.1 kg·m² during the training process.
In this research, sensor noise is added during the training process to improve the robustness of the controller. The attitude and heading reference system (AHRS) used in our flight controller is a mature product, so the attitude angle data and angular velocity data can be read directly from the module. The AHRS module claims that the errors of attitude angle in the roll and pitch axes are between $\pm 0.1^\circ$ and the errors of attitude angle in the yaw axis are between $\pm 1^\circ$. As a result, we add noise conforming to uniform distributions between $\pm 1^\circ$ to attitude angle data in the three axes during the training process.

Table 3 presents the training hyperparameters. The total training process takes approximately 3 h.

### Table 3. Training hyperparameters.

| Parameter                          | Value     |
|------------------------------------|-----------|
| Discount factor $\gamma$           | 0.1       |
| Learning rate of the critic net $\alpha_{\text{critic}}$ | 0.001     |
| Learning rate of the actor net $\alpha_{\text{actor}}$ | 0.001     |
| Batch size $N$                     | 128       |
| Soft update rate $\tau$            | 0.01      |
| Maximum number of agents $n_{\text{agent}}$ | 128       |
| Maximum number of training episodes $e_{\text{max}}$ | 10,000   |
| Control time step $DT$             | 0.01 s    |
| State update time step $dt$        | 0.00001 s |
| Maximum time step in an episode $t_{\text{max}}$ | 5 s       |
| Reward function constant $p_1$     | 1.0       |
| Reward function constant $\epsilon_1$ | 0.01     |
| Reward function constant $\varphi_1$ | $\pi$    |
| Reward function constant $p_2$     | 0.1       |
| Reward function constant $\epsilon_2$ | 0.1      |
| Reward function constant $\varphi_2$ | $2\pi$   |

### 3.4. Performance in Simulation

The RL controller in this article can generate an optimal control law with the MOIs in three axes and the rotor dynamic response constant input. We first test the performance of the plant-protection quadrotor without water in the tank, that is: $J_{xx} = 1.066$, $J_{yy} = 0.961$, $J_{zz} = 1.843$, and $T_m = 0.035$. The performance along the roll axis and yaw axis are shown in Figures 6 and 7, respectively.

![Figure 6. Performance along the roll axis.](image)
The convergence rate of our controller is very high, and nearly no overshoot or steady-state error exists. Next, we test the performance of plant-protection quadrotors with a half-full water tank ($J_{xx} = 1.377$, $J_{yy} = 1.272$, $J_{zz} = 1.908$, and $T_m = 0.035$) and a full water tank ($J_{xx} = 1.540$, $J_{yy} = 1.435$, $J_{zz} = 1.973$, $T_m = 0.035$). The performance along the roll axis is given in Figure 8. We conclude that our controller can adapt to changes in the system parameters and keep the quadrotor flying stably and accurately with any amount of water in the tank.

![Figure 7. Performance along the yaw axis.](image)

![Figure 8. Step response performance along the roll axis.](image)

We test the robustness for sensor noise of the controller in Figure 9. Noises conforming to uniform distributions between $\pm 1^\circ$ are added to attitude angle data of the three axes, the proposed controller still keeps the attitude of the quadrotor around the desired attitude angle, and the attitude errors are less than $1^\circ$, which is acceptable.

Traditional PID controllers are designed for comparison, and their performance and the performance of our RL controller are presented below. As show in Figure 10, PID controller 1 performs well for the quadrotor without water in the tank, but when the tank is filled with water, PID controller 1 shows obvious overshoot compared to our RL controller. As show in Figure 11, PID controller 2 performs well for the quadrotor with a full water tank, but when the tank is empty, PID controller 2 shows a notably lower convergence rate than our RL controller. In conclusion, it is difficult to control the quadrotor well with any amount of water using the same PID parameters, and the performance of our RL controller is much more robust than that of traditional PID controllers with system parameter changes.
In the training process, the states of the quadrotors are initialized randomly, so the quadrotor can recover from any state (even the full turnover state) to zero points. During the training process, the performance of our controller improves gradually. We designed tests to verify these two factors. We test controllers in different episodes (ep = 0, 1000, 5000 and 10,000) on the roll axis of the quadrotor without water in the tank. In Figure 12, the quadrotors recovered to zero point states from random initial states 100 times. We conclude that, after the training process, the controller tends to recover from any state to the zero point and that the performance improves as the training proceeds.
3.5. Flight Controller Platform Details

The flight controller in this project is developed by our team and is called the TensorFly Project; this controller focuses on flight control with RL methods. To meet the demand of real time network calculation, we choose a Raspberry Pi 3B microcomputer (four 1.2 GHz CPU cores) as the main controller in our project. The controller is coded with C++ and calculates matrices using the eigenlibrary \[26\]. The flight controller runs the Raspbian operating system on the board, and we perform substantial work toward reducing the latency of the system. The actor network is calculated using the eigenlibrary \[26\] and, on average, the computation process takes approximately 260 $\mu$s. However, in rare cases, the computation process may take over 1 ms and we think the problem may result from background processes of the onboard operating system. Regardless, the time consumption of the network calculation process is far less than the control period (10 ms), which is acceptable. The AHRS module contained in this controller is a JY901 module, which outputs attitude angle data and angular velocity data in real time directly, and the attitude errors are less than 1.0°. The flight controller and the AHRS module are shown in Figure 13.

Figure 12. Recover tests for the RL controller in different episodes.

(a) RL controller ep = 0.  
(b) RL controller ep = 1000.  
(c) RL controller ep = 5000.  
(d) RL controller ep = 10,000.

Figure 13. Flight controller platform TensorFly Project.
3.6. Flight Performance in Real Flight

The 10 kg class large-load plant-protection quadrotor and its flight using our RL controller are presented in Figure 14.

![Picture of the plant-protection quadrotor.](image1)

![Picture of the quadrotor flying in the air.](image2)

Figure 14. The 10 kg class large-load plant-protection quadrotor in this article.

We test the flight performance of the attitude tracking task on this quadrotor without water and with a half-full water tank, and the performance is presented below. We control the desired attitudes of the quadrotor using a remote controller manually. The flight log of the quadrotor without water in the tank \( J_{xx} = 1.3660, J_{yy} = 1.261, J_{zz} = 2.343, T_m = 0.035 \) is presented in Figure 15, and the attitude errors are presented in Figure 16. The flight log of the quadrotor with a half-full water tank \( J_{xx} = 1.3660, J_{yy} = 1.261, J_{zz} = 2.343, T_m = 0.035 \) is presented in Figure 17, and the attitude errors are presented in Figure 18.

![Graphs](image3)

Figure 15. Attitude tracking performance without water in the tank.
Figure 16. Attitude tracking errors without water in the tank.

Figure 17. Attitude tracking performance with a half-full water tank.

Figure 18. Attitude tracking errors with a half-full water tank.
We conclude that the quadrotor can rapidly and smoothly track the desired attitude (the attitude errors are within 6° during flight), regardless of the amount of water in the tank, which is very practical in real applications.

4. Conclusions

4.1. Discussion

In this article, we present an extended-state RL method for large-load plant-protection quadrotors. We focus on critical problems of plant-protection quadrotors in real applications, such as changes in the moment of inertia during watering work and the slow response rates of large rotors. The RL controller extends these parameters in the state list during training and running, so the controller can automatically adapt to these changes. The introduction of excellent training skills greatly accelerates the training process, and reward shaping skills eliminate steady-state errors. The controller is tested in a simulation environment and applied on a real large-load plant-protection quadrotor platform and the controller achieves great performance regardless of the amounts of water in the tank. Compared to traditional methods that need manual modification and tuning of the controller, our methods show an end-to-end design style, greatly simplify the controller design work and ensure safety in real flight experiments.

4.2. Future Work

During our experimental process, we found that uncertainties such as side winds, movement of the COG of the quadrotor, wobbling water in the tank and external disturbances can increase the flight stability. Active disturbance rejection control (ADRC) methods will be applied and tested in our future work to solve these practical problems for large-load plant-protection quadrotors.

The extended-state RL method proposed in this paper can be widely used on several other platforms such as unmanned underwater vehicles and robot arms. We note that the DAI methods can achieve a similar performance, and we plan to consider this comparative methodology to further improve the performance of our methods.

Author Contributions: Conceptualization, D.H., Z.P. and Z.T.; methodology, D.H., Z.P. and Z.T.; software, D.H.; validation, Z.P.; formal analysis, D.H.; investigation, D.H.; resources, Z.P. and Z.T.; data curation, D.H.; writing—original draft preparation, D.H.; writing—review and editing, D.H., Z.P. and Z.T.; supervision, Z.P.; project administration, D.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References
1. Chovancova, A.; Fico, T.; Duchon, F.; Dekan, M.; Chovanec, L.; Dekanova, M. Control Methods Comparison for the Real Quadrotor on an Innovative Test Stand. Appl. Sci. 2020, 10, 2064.
2. Anderson, C. 10 Breakthrough Technologies: Agricultural Drones. MIT Technol. Rev. 2014, 3, 58–60.
3. PX4, Group. PX4: A Professional Open Source Autopilot Stack. Available online: https://github.com/PX4/Firmware (accessed on 27 March 2021).
4. ArduPilot Group. The ArduPilot Project Provides an Advanced, Full-Featured and Reliable Open Source Autopilot Software System. Available online: http://ardupilot.org/ (accessed on 27 March 2021).
5. Yu, Y.; Yang, S.; Wang, M.; Li, C.; Li, Z. High performance full attitude control of a quadrotor on SO(3). In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015.
6. Jafari, H.; Zareh, M.; Roshanian, J.; Nikkhah, A. An Optimal Guidance Law Applied to Quadrotor Using LQR Method. Trans. Jpn. Soc. Aeronaust. Space Sci. 2010, 53, 32–39.
7. Falanga, D.; Kleber, K.; Mintchev, S.; Floreano, D.; Scaramuzza, D. The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly. *IEEE Robot. Autom. Lett.* **2019**, *4*, 209–216.

8. Bouabdallah, S.; Siegwart, R. Full control of a quadrotor. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007.

9. Bouabdallah, S.; Siegwart, R. Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005.

10. Doukhi, O.; Fayjie, A.; Lee, D.J. Global fast terminal sliding mode control for quadrotor UAV. In Proceedings of the 2017 17th International Conference on Control, Automation and Systems (ICCAS), Jeju, Korea, 18–21 October 2017; pp. 1180–1182.

11. Noormohammadi-Asl, A.; Esrafilian, O.; Arzati, M.A.; Taghirad, H.D. System identification and H-infinity-based control of quadrotor attitude. *Mech. Syst. Signal Process.* **2020**, *135*, 106358.

12. Liang, X.H.; Wang, Q.; Hu, C.H.; Dong, C.Y. Observer-based H-infinity fault-tolerant attitude control for satellite with actuator and sensor faults. *Aerosp. Sci. Technol.* **2019**, *95*, 105424.

13. He, Z.; Gao, W.; He, X.; Wang, M.; Liu, Y.; Song, Y.; An, Z. Fuzzy intelligent control method for improving flight attitude stability of plant protection quadrotor UAV. *Int. J. Agric. Biol. Eng.* **2019**, *12*, 110–115.

14. Sands, T. Development of Deterministic Artificial Intelligence for Unmanned Underwater Vehicles (UUV). *J. Mar. Sci. Eng.* **2020**, *8*, 578.

15. Sands, T. Optimization Provenance of Whiplash Compensation for Flexible Space Robotics. *Aerospace* **2019**, *6*, 93.

16. Jemin, H.; Inkyu, S.; Roland, S.; Marco, H. Control of a Quadrotor with Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2017**, *4*, 2096–2103.

17. Lin, X.N.; Yu, Y.; Sun, C.Y. Supplementary Reinforcement Learning Controller Designed for Quadrotor UAVs. *IEEE Access* **2019**, *7*, 26422–26431.

18. Ma, H.-J.; Xu, L.-X.; Yang, G.-H. Multiple Environment Integral Reinforcement Learning-Based Fault-Tolerant Control for Affine Nonlinear Systems. *IEEE Trans. Cybern.* **2021**, *51*, 1913–1928.

19. Lin, X.; Liu, J.; Yu, Y.; Sun, C. Event-triggered reinforcement learning control for the quadrotor UAV with actuator saturation ms. *Neurocomputing* **2020**, *415*, 135–145.

20. Hu, D.; Pei, Z.; Tang, Z. Single-Parameter-Tuned Attitude Control for Quadrotor with Unknown Disturbance. *Appl. Sci.* **2020**, *10*, 5564.

21. Lillicra, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *Computerence* **2015**, doi:10.1016/S1098-3015(10)67722-4.

22. Shi, D.J.; Dai, X.H.; Zhang, X.W.; Quan, Q. A practical performance evaluation method for electric multicopters. *IEEE/ASME Trans. Mechatron.* **2017**, *3*, 1337–1348.

23. Diederek, P.K.; Jimmy, L.B. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.

24. Wang, Y.; Sun, J.; He, H.B.; Sun, C.Y. Deterministic Policy Gradient with Integral Compensator for Robust Quadrotor Control. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *50*, 3713–3725, doi:10.1109/TSMC.2018.2884725.

25. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, G.M.; Graves, A.; Riedmiller, M.; Andreas, K.A.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2019**, *518*, 529–533.

26. Eigen Group. Eigen Is a C++ Template Library for Linear Algebra: Matrices, Vectors, Numerical Solvers, and Related Algorithms. Available online: https://eigen.tuxfamily.org/ (accessed on 27 March 2021).