APER-DDQN: UAV Precise Airdrop Method Based on Deep Reinforcement Learning

YAN OUYANG, XINQING WANG, RUIZHE HU, AND HONGHUI XU

Department of Mechanical Engineering, College of Field Engineering, Army Engineering University, Nanjing 210007, China
Corresponding author: Xinqing Wang (wangxinqing510@163.com)

This work was supported in part by the China National Key Research and Development Program under Grant 2016YFC0802904, in part by the National Natural Science Foundation of China under Grant 61671470, and in part by the 62nd batch of funded projects of China Postdoctoral Science Foundation under Grant 2017M623423.

ABSTRACT  Accuracy is the most critical factor that affects the effect of unmanned aerial vehicle (UAV) airdrop. The method to improve the accuracy of UAV airdrop based on traditional modeling has some limitations such as complex modeling, multiple model parameters and difficulty in considering all kinds of factors comprehensively when facing complex realistic environment. In order to solve the problem of UAV precision airdrop more conveniently, this paper introduces the deep reinforcement learning method and proposes an Adaptive Priority Experience Replay Deep Double Q-Network (APER-DDQN) algorithm based on Deep Double Q-Network (DDQN). This method introduces the priority experience replay mechanism based on DDQN, and adopts adaptive discount rate and learning rate to improve the decision-making performance and stability of the algorithm. Furthermore, this paper designs and builds a simulation experimental platform for algorithm training and testing. The experimental results show that our APER-DDQN has good performance and can more effectively solve the problem of UAV accurate airdrop while avoiding the complex modeling process. Firstly, in the training stage, compared with DDQN and Deep Q Network (DQN), APER-DDQN has faster convergence speed, higher reward and more stable performance. Then, in the test phase, compared with relying on human experience, our method shows higher average reward (average 3.01) and success rate (average 41%), and our method also has more advantages in performance compared with DDQN and DQN. Finally, extended experiments verify the generalization ability of APER-DDQN to different environments.

INDEX TERMS  UAV airdrop, deep reinforcement learning, double deep Q-network, priority experience replay.

I. INTRODUCTION  Unmanned aerial vehicle (UAV) has the characteristics of high mobility, low cost, small size, flexible control, and autonomous flight [1], which can adapt to various natural environments. In recent years, with the development and maturity of control technology and electronic technology, the performance of UAV has been rapidly improved in all aspects. UAV has been applied to various fields such as military, civil and scientific and technological [2], and is playing a more and more important role in these fields.

UAV airdrop is one of the many application fields of UAV. As shown in Figure 1, UAV airdrop can be used for marine rescue, material delivery and bomb dropping. And in these applications, the accuracy of dropping is very important.

The associate editor coordinating the review of this manuscript and approving it for publication was Emre Koyuncu.

However, in complex rescue or combat environments, due to the interference of strong winds, the UAV sways seriously.
Based on the above considerations, this paper proposes a scheme to solve the problem of autonomous and accurate airdrop of UAV through deep reinforcement learning method. Specifically, we first designed and built an experimental platform for simulating the state of UAV airdrop, through which we can collect a large amount of sample data for training and testing of deep reinforcement networks. Then for practical problems, we designed an improved deep reinforcement learning algorithm on the basis of DDQN [6], called Adaptive Prioritized Experience Replay DDQN (APER-DDQN), which utilizes the powerful learning and decision-making ability of deep reinforcement algorithms to achieve precise airdrop of UAV. Finally, we test the trained algorithm on our experimental platform. The experimental results show that our proposed scheme can effectively solve the problem of accurate airdrop of UAV.

The main contributions of this paper are summarized as follows:

1) The deep reinforcement learning method is introduced into UAV airdrop for the first time, and a UAV airdrop decision model based on improved deep reinforcement learning algorithm is designed;

2) Design and build an experimental platform that simulates the state of UAV airdrop for training and testing our decision-making model;

3) In order to efficiently screen and use training samples, we introduce the priority experience replay mechanism into DDQN algorithm. At the same time, we use adaptive discount rate and learning rate to improve the training convergence speed and network stability.

4) Comparative experiments verify the effectiveness of our designed APER-DDQN method to solve the problem of autonomous and precise airdrop of UAV. Compared with DDQN and DQN on the experimental platform, our algorithm has faster training convergence, higher reward, and higher stability. Compared with relying on human experience decision-making, our method greatly improves the accuracy of airdrops, and both the average reward and the success rate have obvious advantages.

The remainder of this paper is organized as follows: Section 2 reviews related work on deep reinforcement learning and precision airdrops. Section 3 introduces the construction process of APER-DDQN decision model in detail. Section 4 presents and analyzes the results of the comparative and generalization experiments. Finally, section 5 summarizes the full text and puts forward the prospect of future work.

II. RELATED WORK

A. REINFORCEMENT LEARNING

Reinforcement learning (RL), as a research hotspot in the field of machine learning, can be applied to robot control [7], simulation [8], industrial manufacturing [9], game gaming [10], optimization and scheduling [11] and autonomous driving [12] and other fields. The basic idea of RL is that in the process of interacting with the environment, the
agent (agent) continuously adjusts its own strategy according to the reward obtained by the environment feedback to achieve the best decision. The basic elements of RL include Policy, Reward Function, Value Function and Environment [13]. Markov Decision Process (MDP) can be used to model the RL problems [14]. Supervised learning requires large-scale datasets and labels to learn a classifier or distribution of data points. Compared with supervised learning algorithms, the advantage of RL methods is that they do not require labels to indicate the target class, and it can obtain the data required for training by interacting with the environment.

B. DEEP REINFORCEMENT LEARNING

Deep learning (DL) has powerful perception and expression capabilities, but is not good at decision optimization. Although traditional RL can effectively solve decision optimization problems, it is only suitable for relatively simple scenarios. When facing complex practical problems, traditional table RL is no longer applicable. Thus the combination of DL and RL, namely Deep Reinforcement Learning (DRL), has become an effective way to solve optimal decision-making problems in complex real-world scenarios. Lange et al. first proposed a Deep Auto-Encoder (DAE) model by combining DL models and RL methods [15]. However, the model is only proved to be suitable for control problems with small dimension of state space, such as lattice world task based on visual perception. Lange et al. further proposed the Deep Fitted Q-learning (DFQ) algorithm [16], and applied the algorithm to vehicle control. In the Google DeepMind team, Mnih et al. combined the convolutional neural network with the Q-learning [17] algorithm in traditional RL, proposed a Deep Q Network (DQN) model [5], [18], and did a lot of experiments on the game platform based on Atari2600. The results show that in most games, the game performance of DQN has caught up or even exceeded the level of human players. In view of the overestimation problem of the DQN algorithm, Van Hasselt et al. proposed the Deep Double Q-Network (DDQN) [6] based on the research of the Double Q learning algorithm [19]. When training DDQN, Schaul et al. replaced the equal-probability sampling method with a priority-based experience replay mechanism [20], which improved the utilization of valuable samples and enabled the agent to achieve higher scores in some Atari games. Francois-Lavet et al. used adaptive discount factor and learning rate in DQN [21], which accelerated the speed of deep network convergence. DRQN [22] proposed by Hausknecht et al. uses a recurrent neural network to memorize continuous historical state information on a timeline. Lillicrap et al. proposed a deep deterministic policy gradient (DDPG) algorithm [23] based on AC framework to solve the DRL problem on the space of continuous action. Mnih et al. proposed the Asynchronous Advantage Actor-Critic (A3C) based on the idea of asynchronous reinforcement learning [24].

C. PRECISE AIRDROP

Airdrop is a work with huge application value. Whether it is in the military or civilian fields, airdrop plays an irreplaceable role. At the same time, airdrop is also a very challenging work. In the face of complex and changeable air environment, it is difficult to accurately airdrop the dropped object to the target position. For the sake of improving the accuracy of airdrop, a lot of research work and practice have been carried out. Continuously computed impact point (CCIP) aims at the bombing target by continuously predicting the location of the impact point. Continuously computed release point (CCRP) accurately hits the target by constantly predicting the location of the bomb drop point [25]. [26] improved the accuracy of airdrops by dynamically modeling the parachute dynamics and stability of the airdrop system. [27] presented the Joint Precision Airdrop System project led by the US military. [28] studied precision airdrop control via satellite and inertial navigation. [29] improved the operating accuracy of the system by modeling and simulating a 9 degree of freedom system for a remote-controlled airdrop system. [30] established a simplified simulation model of the airdrop trajectory, and analyzed the impact of different initial conditions on the airdrop accuracy. Although these research works have made significant contributions to improving the accuracy of airdrops, these methods all have some shortcomings and limitations. With the continuous development and maturity of DRL, it is increasingly common to use DRL methods to solve decision optimization problems. In this paper, DRL is introduced into precise airdrops to solve the decision control problems in airdrops.

III. AIRDROP MODEL DESIGN BASED ON APER-DDQN ALGORITHM

In this section, we will introduce the specific design ideas and practical processes of the airdrop decision model based on the APER-DDQN algorithm. Figure 2 shows the overall framework of this paper. We first set up the experimental platform, which includes three major parts: airdrop principal part, airdrop area part and control terminal part. Then the constituents of the MDP are instantiated, the most important part of which is the design of the reward function. Finally, the airdrop decision model based on the APER-DDQN is trained and tested on our experimental platform. Specifically, Section 3.1 introduces the design and construction of the experimental platform, Section 3.2 describes the Markov decision process of the airdrop problem, and Section 3.3 introduces the APER-DDQN algorithm in detail.

A. DESIGN AND CONSTRUCTION OF EXPERIMENTAL PLATFORM

The method of acquiring training data by using drones on the spot is too inefficient, and the cost of acquiring data is huge, which cannot meet the sample requirements for training deep reinforcement networks. Although the method of obtaining training data by computer-simulated airdrops can obtain a
large number of samples at low cost, the simulated environment is too standardized and cannot include various influencing factors in real airdrops. Therefore, we solve the problem of obtaining training samples by building a small simulated airdrop physical platform. The physical simulation platform we built is shown in Figure 3. The whole simulation platform is grouped into three parts, including airdrop principal part, airdrop area part and control terminal part.

The principal part of our airdrop is the execution part of the airdrop experiment, which is mainly used to simulate the state of UAV in the natural environment, perform airdrop action, collect and transmit experimental samples. As shown in Figure 4, the principal part of our airdrop is composed of component power supply (a 15000mAh battery), Jetson TX2 computing processing unit, four serial port relays, camera unit, motor, power-off electromagnet and aluminum alloy square tube frame and other components. Through
independent design, we reasonably drill holes on the aluminum alloy square tube frame, and use bolts and nuts to fasten the components to the frame. Component power supplies are mainly used to provide 12V power to motors, relays and electromagnets. The Jetson TX2 is a small computer module that can be deployed to UAV, and we use it to capture and process the video frames we need. The four serial port relay is used to comprehensively control the opening and closing of the power supply of the motor and electromagnet. The relay itself is controlled by the program written in TX2. The camera unit is mainly used to capture the airdrop status in the form of video and transmit it to TX2 for processing. The motor is used to realize the automatic recovery of the simulated projectiles and the shaking control of the square tube. Electromagnets are used to attract our magnetic simulated projectiles. The main line connections of the principal part is shown in Figure 5.

The airdrop area is made of 1.2m × 1.2m square resin board. For the sake of reducing the difficulty of the experiment and improving the identification of the airdrop area, we wrapped the resin board with a white flannel and marked the center of the target area. At the same time, in order to facilitate RL training, we design a schematic diagram of the area division of the airdrop score. As shown in Figure 6, we divide the possible drop areas of all samples into five levels, and the areas of different levels in the figure are distinguished by different colors. \(D_c\) is the distance from the center of the target area to the center of the projectile, \(r\) is the radius of the projectile. Taking the center of the target as the center of the circle, the white circular area with a radius of 0.2\(r\) is the first level \(R_1\), and the score (i.e. the reward) of the projectile center falling on this area is the highest, which is 10 points. The yellow annular area with a radius greater than 0.2\(r\) and less than 2.2\(r\) is the second level \(R_2\), and the score for falling into this area is 6 points. The blue annular area with a radius greater than 2.2\(r\) and less than 6.2\(r\) is the third level \(R_3\), and the score for falling into this area is 3 points. The green annular area with a radius greater than 6.2\(r\) and less than 12.2\(r\) is the fourth grade \(R_4\), and the score for falling into this area is 1 point. Other areas with a radius greater than 12.2\(r\) are the fifth grade \(R_5\), and the center of the projectile falls into this area with a score of 0.

The control terminal part is mainly used to realize the preprocessing of training samples and the sending of decision signals. We use ROS distributed programming to process the collected samples and synchronously control the electromagnets and motors.

### B. MARKOV DECISION PROCESS FOR THE AIRDROP PROBLEM

In this section, we transform the airdrop decision problem into a Markov Decision Process (MDP). The basic elements of MDP include states, actions, and reward functions, usually denoted as \((S, A, R)\) [31], where \(S\) represents the set of all environmental states, including all states of the agent, \(A\) represents the executable action space, including All actions that the agent can take, \(R : S \times A \rightarrow R\) represents the reward function. At each time step \(t\), the agent observes the environment state \(s_t \in S\), and then select an action \(a_t \in A(s)\) based on the policy \(\pi\) [31], \(A(s)\) represents the set of all actions \(a_t\) that the agent can perform in state \(s_t\). After obtaining the reward \(r(s, a)\) from the environmental feedback, the agent moves to the next state \(s_{t+1} \in S\). In this paper, the agent is the controller of projectile, and the environment is the airdrop experimental platform. The interaction process between the environment and the agent is shown in Figure 7. The constituent elements of MDP are instantiated as follows.

1) **STATE**

State \(S\) contains a set of motion state information of the simulated projectile during the sloshing process, including the speed of the simulated projectile, the sloshing direction and the spatial position relative to the center of the target. This information is provided by the video frames captured by the experimental platform, and we take the captured four video frames as the direct state input of the deep reinforcement model.

2) **ACTION**

The agent makes control decisions according to the observed state information and chooses to take appropriate actions. There are two kinds of actions in our model: throwing and not throwing. We use a binary variable \(a\) to represent the action in the model, \(a = 1\) means throwing, and \(a = 0\) means not throwing.

3) **REWARD FUNCTION**

We take the center of the target area as the center of the circle, and divide our circular scoring area with different lengths as the radius. As shown in Figure 6, when the center of the simulated projectile falls into the area of the corresponding color, the agent will get a score reward matching the area. The goal of our agent is to make the simulated projectile hit the target area as accurately as possible, that is, the throwing score is the highest. Therefore, we define the reward function
At the same time, the samples we use to train the network are pictures taken by the camera. In order to establish the mapping relationship between the drop point map and the reward, we use a simple convolutional neural network to construct the reward mapping function. As shown in Figure 8, our reward mapping network consists of three convolutional layers (with ReLU activation function) and two fully connected layers. We removed the pooling layer used in general convolutional networks because the pooling layer makes the network insensitive to the position of objects in the image, and the position information of the projectile and the target area in the image is a key factor in determining the reward. After sufficient training, this network can achieve a one-to-one mapping of drop points to rewards.

From the point of view of MDP, from state $s_t$, select action $a_t$ reaches the next state $s_{t+1}$, this process is determined by the policy $\pi$, through which the agent interacts with the environment. State-action value function $Q_\pi(s, a)$ is employed for evaluating the quality of throwing action in a given state:

$$Q_\pi(s, a) = E_\pi[ R_t | s_t = s, a_t = a ]$$

where $\gamma \in [0, 1]$ is the discount rate [32] that expresses the importance of future rewards. The ultimate goal of MDP is to find the optimal control strategy $\pi^*$ to maximize the state-action value function $Q_\pi(s, a)$, that is, the optimal state-action value function $Q_{\pi^*}(s, a)$, which can be expressed as:

$$Q_{\pi^*}(s, a) = \max_\pi E [ R_t | s_t = s, a_t = a ]$$

(3)

The optimal state-action value function follows the Bellman optimality equation [33]:

$$Q_{\pi^*}(s, a) = E_{s' \sim S} \left[ r + \gamma \max_{a'} Q_{\pi^*}(s', a') | s, a \right]$$

(4)

For the solution of the optimal state-action value function, the traditional RL algorithm is implemented by iterating the Bellman equation:

$$Q_{i+1}(s, a) = E \left[ r + \gamma \max_{a'} Q_i(s', a') | s, a \right]$$

(5)

$r_t$ of the agent at time step $t$ as follows:

$$r_t(s_t, a_t) = p, \quad (p \in \{0, 1, 3, 6, 10\}, s_t \in S, a_t \in A) \quad (1)$$

FIGURE 6. Schematic diagram of the reward design. (a) Delineate the reward area according to the center distance between the target area and the throwing object ($D_c$), (b) Determine the reward level and score according to the position of the center of the throwing object. Where $r$ is the radius of the projectile, $D_c$ represents the center distance between the target area and the projectile, and the radius of the target area is 1.2 $r$. 

FIGURE 7. The interaction of the agent with the environment.
However, in the airdrop decision problem studied in this paper, the state of the agent (such as the velocity of the projectile) is continuous, and the solution based on the traditional RL algorithm needs to construct a huge table to approximate the state-action value function. Updating such a table is computationally expensive. In response to this challenge, studies have shown that introducing DL into RL and using DL to approximate the Q-value function is an effective solution. Therefore, this paper adopts DRL to achieve the control decision of airdrop.

Our action space contains only two types of actions and is a discrete variable, thus we need a DRL algorithm suitable for application in continuous state space and discrete action space to make control decisions for our airdrop model. Specifically, we choose the DDQN algorithm proposed by Van Hasselt et al. [6] as the basic framework of the whole research. Compared with traditional RL algorithms, DDQN has the following three major improvements: at first, DDQN is a kind of DRL algorithm, which employs a deep neural network to approximate the state-action value function, second, DDQN adopts an experience replay mechanism [34] during training to reduce the correlation between samples, which improves the stability of the algorithm, third, DDQN uses two neural networks for action selection and strategy evaluation [31] respectively, which reduces the risk of overestimating the Q value [35].

C. AIRDROP DECISION-MAKING METHOD BASED ON APER-DDQN

Traditional DDQN algorithms employ an experience replay mechanism [34] to store and use training samples. During the training period, the experience replay mechanism extracts mini-batches of samples equiprobably from the sample pool for training. This uniform sampling method does not consider the importance of each sample. To address this issue, we adopt the Prioritized Experience Replay (PER) [20] mechanism proposed by Schaul et al. to screen and use efficient training samples. Meanwhile, we adopt an adaptive discount rate and learning rate [21] to train our network for speeding up the convergence speed of the network and improving the stability of the algorithm. The airdrop decision method framework based on APER-DDQN is shown in Figure 9. In the whole airdrop decision-making framework, the input is four frames of images recording the state of the simulated projectile, and the output is the estimated Q value of each action.

In DDQN, action selection and evaluation are separated from each other. It uses the current Q network to select the optimal action. The Q value output by the current Q network can be expressed as

$$Q(s_t, a_t | \theta)$$

where \( \theta \) is the weight parameter of the current Q network; The target network is employed for evaluating the selected action. The Q value of the target network can be expressed as:

$$y_t = r_t + \gamma Q(s_{t+1}, \arg \max_a Q(s_{t+1}, a | \theta) | \theta^-)$$

where \( \theta^- \) is the weight parameter of the target network, and the parameter \( \theta \) is copied to \( \theta^- \) from the current Q network every fixed number of steps. The main goal of DDQN is to minimize the mean squared error between the current Q value and the target Q value. The error function is:

$$L(\theta) = E \left[ (y_t - Q(s_t, a_t | \theta))^2 \right]$$

In the PER mechanism [20], the time difference (TD) error [36] term \( \mu_i \) of each sample \( e_i \) is used to evaluate the priority of the samples. The time difference error term is defined as:

$$\mu_i = y_t - Q(s_t, a_t | \theta)$$

where \( y_t \) is calculated as follows:

$$y_t = r_t + \gamma Q(s_{t+1}, \arg \max_a Q(s_{t+1}, a | \theta) | \theta^-)$$

FIGURE 8. The structure of the reward mapping network.
During the training process, the probability $P(i)$ of sample $e_i$ being drawn is defined as:

$$P(i) = \frac{p_i^\alpha}{\sum_{k=1}^{N} p_k^\alpha}$$  \hspace{1cm} (10)

where $p_i = |\mu_i| + \epsilon$ represents the priority of sample $e_i$, and $\epsilon$ is a small positive constant to avoid edge cases when the time difference error term is 0. $\alpha$ determines the degree to which precedence is used.

Prioritized experience replay [20] introduces bias, which needs to be corrected by importance sampling (IS) weights [37]:

$$\omega_i = \frac{(M_b p_i)^{-\beta}}{\max_j \omega_j}$$  \hspace{1cm} (11)

where $M_b$ is the size of the sample pool, and the parameter $\beta$ is a positive constant.

At each epoch of training, we use a gradually increasing discount rate $\gamma_n$ to adapt our training process:

$$\gamma_{n+1} = 1 - 0.98 (1 - \gamma_n)$$  \hspace{1cm} (12)

Meanwhile, in order to enhance the stability of the system, we modify the learning rate $\eta$ as follows:

$$\eta_{n+1} = 0.98 \eta_n$$  \hspace{1cm} (13)

Therefore, the error function of the improved DDQN is updated as:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \omega_i (r_i + \gamma_n \max_a Q(s_{i+1}, a|\theta) - Q(s_i, a_i|\theta))^2$$  \hspace{1cm} (14)

The gradient ascent algorithm is employed for updating the network parameters:

$$\theta \leftarrow \theta + \eta_n \cdot \nabla_{\theta} L(\theta)$$  \hspace{1cm} (15)

where $\eta_n$ is the adaptive learning rate.

We train our APER-DDQN algorithm offline by taking the collected data as the environment state. After the algorithm is trained, the parameters of the network are fixed for the control decision of our airdrop platform. The training process of the airdrop decision network of APER-DDQN is shown in Algorithm A1.

IV. EXPERIMENTS AND ANALYSIS

In this section, we evaluate the performance of our proposed APER-DDQN method and the practical effect of solving the airdrop problem through comparative experiments. First,
we introduce the preparation, running environment and network structure parameters of the experiment. Then, compared with DDQN and DQN, we analyze and compare the training results of APER-DDQN. At the same time, after the training is stable, we test the above three methods and rely on artificial experience for decision-making, and analyze the experimental results. Finally, we conduct extended experiments to evaluate the generalization ability of APER-DDQN.

A. EXPERIMENTAL SETUP

1) PREPROCESSING

Whether it is a reward mapping network or an APER-DDQN network, their training requires images as input data, and we use the built experimental platform to collect the data required for network training. The principal part of the airdrop is dropped in the random initial state, and the state of the simulated dropped object is recorded by the camera on the square tube in the form of video and transmitted to the control terminal. The program written by the control terminal extracts every two frames from the video, and extracts a total of four frames of images to record the state of an airdrop. The size of the extracted original image is 800 × 600. If the original image is directly employed for the input of network training, the calculation and storage cost of processing data is too high. So we generate thumbnails with a corresponding scale of 160 × 120 by down-sampling. At the same time, remove some worthless pixels on the border, and crop the thumbnail to 120 × 120 size. Then the cropped image is grayscaled to reduce the computational cost. At the same time, the landing result of an airdrop is also recorded by the camera and transmitted to the control terminal for the above-mentioned preprocessing, which is employed for the training input data of the reward mapping network.

2) EXPERIMENT RUNNING ENVIRONMENT

Network training was performed on a computer with an Intel Core i7-6700 CPU and a GeForce RTX 3090 GPU. The operating environment is cuDNN 7.5.0, CUDA 10.0, TensorFlow 2.0, Python version 3.7, and the operating system is Ubuntu 18.04.5 LTS.

3) NETWORK STRUCTURE AND HYPERPARAMETERS

Our APER-DDQN consists of three convolutional layers and two fully connected layers. The first convolutional layer contains 32 filters of size 8 × 8 with stride 4. The second layer contains 64 filters of size 5 × 5 with stride 3. The third layer contains 64 filters of size 3 × 3 with stride 1. Each convolutional layer uses a linear rectifier unit (ReLU) as the activation function. The outputs of the two fully connected layers are 512 and 2, respectively. For parameter selection, we refer to the common practice in the DL community [38], and make repeated experiments and adjustments according to the actual situation of our application. The initial learning rate \( \eta = 0.001 \), and the Adam optimizer is used to update the network weight. The initial discount rate \( \gamma = 0.9 \), and the mini-batch size is 32. The learning rate and discount rate are adaptively changed at each epoch of training according to equations (13) and (15). The parameters of the improved priority experience replay pool are: \( \alpha = 0.5, \beta = 0.6, M_b = 15000 \). Meanwhile, our reward mapping network takes on almost the same network structure as APER-DDQN, the only difference is that the output of the last fully connected layer is 1, and adopts stochastic gradient descent [39] method.

---

Algorithm A1 Training Procedure of Airdrop Decision Network Based on APER-DDQN

1: initialization: random weights for Q-network (\( \theta \)) and target network (\( \theta^- \longrightarrow \theta \))
2: Initialization: prioritized experience replay buffer, mini-batch size, hyper-parameters
3: for episode = 1 to M do
4:   Receive the initial state \( s_1 \) (four frame in image) of the environment from the airdrop simulation platform
5:   for \( t = 1 \) to \( T \) do
6:     Select an action \( a_t \) at a given state \( s_t \) using \( \epsilon \)-greedy policy
7:     Execute action \( a_t \), transit to the next state \( s_{t+1} \) and observe the reward \( r_t \)
8:     Store transition \((s_t, a_t, r_t, s_{t+1})\) in the prioritized experience replay buffer and set \( p_t = \max_{i < t} p_i \)
9:     for \( i = 1 \) to \( N \) do
10:        Sample transition \( i \) with probability \( P_i \) in (10)
11:        Compute the importance-sampling weights \( \omega_i \) using equation (11)
12:        Compute the absolute TD-error \( |\mu_i| \) using equation (8)
13:        Update transition priority \( p_i \) according to \(|\mu_i|\)
14:     end for
15:     Estimate the target \( y_i \)
16:     Update weights \( \theta \) using (15)
17:     Copy weights into the target network (\( \theta^- \)) every fixed number of steps
18:   end for
19:   Update discount factor \( \gamma \) and step-size \( \eta \) using equation (12) and (13)
20: end for
to update network parameters. The initial learning rate is 0.01, which is reduced by a factor of 10 for every 10 epochs of training, for a total of 50 epochs of training, and the mini-batch size is 32.

4) EXPERIMENTAL PLATFORM DESCRIPTION
Our simulation experiment platform is built based on aluminum alloy profile frame, a total of 12 square tubes with the size of 1.2m × 1.2m × 2m, which constitutes the complete space of UAV airdrop system. The main body of the platform is composed of a hollow square shaft. In order to facilitate the simulation of shaking, the material is light weight aluminum alloy, with a size of 0.06m × 0.06m × 0.5m and a wall thickness of 3mm. The square shaft is suspended under a steel square tube by means of a ball bearing connection and can be rotated freely. A roller is welded on each side of the lower surface of the steel pipe and embedded in the aluminum profile square pipe on both sides of the frame, so that the steel pipe can slide along the groove and drive the platform movement. Because this solution focuses on simulating the swing of the UAV, the six physical quantities (longitude, latitude, altitude, pitch, roll, and deflection angle) required to describe the UAV’s flight state are simplified to one degree of translational freedom and one degree of rotational freedom.

In the experiment, the UAV power system is simulated by an electric pusher, and the sliding of the steel pipe is pushed, and the speed is set to 0.05m/s by default. The diversity of attitude is simulated by ball bearing. Due to the limitation of conditions, the acceleration is not considered temporarily.

B. TRAINING RESULTS
During the training process, we define 10 airdrops as a training episode, and train a total of 300 episodes. In the training of each new episode, the initial sloshing angle is randomly selected in the range of [-60°, 60°]. Considering the diversity of UAV attitude, the initial disturbance is manually added to each airdrop to randomly change its shaking direction. We use the sum of the rewards obtained in each training episode as our performance evaluation metric. For the sake of the verification about the effectiveness of the APER-DDQN decision model, we compared our APER-DDQN algorithm with the DQN algorithm and the DDQN algorithm in the same environment for its decision-making ability. We use the same network structure and software and hardware platforms for training.

As shown in Figure 10(a), compared with the other two DRL methods, our APER-DDQN method has a faster rising speed of the training curve, indicating that our APER-DDQN method has the highest learning efficiency. In our method, when training about 50 episodes, the reward gradually tends to be stable, while DDQN needs to be trained for about 100 episodes, and DQN needs more time, about 120 episodes. Meanwhile, it can be told by the training curve in Figure 10(a) that the APER-DDQN algorithm can finally achieve a higher reward, while the rewards of DDQN and DQN are relatively lower. After training 200 episodes, the reward of using APER-DDQN method can be maintained above 80 and relatively stable, while the reward of the other two methods is low and fluctuates greatly, which shows that our APER-DDQN method not only has better decision-making effect for our airdrop problem, but also is more stable. Figure 10(b) reveals the advantages of the APER-DDQN method more concretely. From Figure 10(b) it can be told that the average reward of 300 episodes trained by APER-DDQN method is 72.13, while the average reward values of DDQN and DQN are 59.36 and 50.44 respectively, which also proves that our APER-DDQN has stronger decision-making ability.

C. TEST RESULTS
For the sake of the verification about the actual decision-making performance of the APER-DDQN method, for the airdrop problem, we test the APER-DDQN after training is stabilized. At the same time, in order to facilitate the demonstration of the comparative effect, we also conducted comparative tests on the three cases of DDQN after training, DQN after training and decision-making by manual experience. The tests were uniformly implemented on the simulation platform we built, and a total of 90 airdrop tests were carried out. The 90 tests were divided into three different
initial conditions: the initial sloshing angle of 60° (60°), the initial sloshing angle of 30° (30°), and the random initial angle (Random), each of which was tested 30 times. We use the average reward and success rate of 30 tests to evaluate test performance. We define that the center of the throwing object falls within the range of $R_1$ and $R_2$ as success, that is, the reward of a single throw is not less than 6, which is considered a success. Higher average rewards and success rates represent better decision making. Figure 11 is a sample of selected representative test results. Figure 12-14 shows the reward distribution of 30 test results under the conditions of sloshing initial angle of 60°, 30° and random initial angle respectively. From the test result example in Figure 11 and the reward distribution in Figure 12-14, it can be seen intuitively that the trained APER-DDQN has superior decision-making performance compared with other methods.

The test results are shown in Table 1. After training, our APER-DDQN can achieve a quite good airdrop decision-making effect. The average rewards of APER-DDQN were 7.20 (60°), 7.07 (30°) and 7.63 (random) respectively. And among several other decision-making methods, the largest average reward is only 6.30 (DDQN). In the tests
of three different initial conditions, the average reward of our APER-DDQN is much higher than other decision methods. At the same time, the success rate of APER-DDQN also reached more than 0.8, which were 0.87 (60°), 0.83 (30°) and 0.9 (random). The success rates of other methods are all below 0.8. When the initial sloshing angle is 30° and a random initial angle, the decision-making based on artificial experience has even only a success rate of 0.43. These test results indicate that our APER-DDQN method has good performance advantages compared with other methods in solving the airdrop decision-making problem. The reason is that APER-DDQN not only introduces the priority experience replay mechanism, but also improves the discount rate and learning rate of the algorithm. Experiments show that these improvements can more efficient training samples and obtain a stable training model. In particular, compared with the decision-making based on human experience, our APER-DDQN method has an advantage of more

| Method         | 60°  | 30°  | Random |
|----------------|------|------|--------|
|                | AR   | SR   | AR     | SR   |
| DDQN           | 6.17 | 0.77 | 5.90   | 0.67 |
| DQN            | 4.93 | 0.53 | 4.90   | 0.57 |
| Manual operation DQN | 4.57 | 0.50 | 4.20   | 0.43 |
| APER-DDQN      | 7.20 | 0.87 | 7.07   | 0.83 |
|                |      |      | 7.63   | 0.90 |
than 2.5 points in average reward and at least 0.35 higher success rate in either case, indicating that APER-DDQN can effectively solve the difficulty of accurate airdrop decision-making.

### D. EXTENDED EXPERIMENTS

For the sake of the further verification about the generalization ability of APER-DDQN, we designed extended experiments to test the performance of APER-DDQN by throwing from different directions, heights and points. For the convenience of comparison, we unified the initial sloshing angle of the test to 60°. In order to make the test results robust, we tested 30 times in different environments, and also used the average reward and success rate of the 30 times as the evaluation standard. The test results are shown in Figure 15 and Table 2.

It can be found intuitively from Figure 15 and table 2 that, on the whole, the results of airdrop decision-making through APER-DDQN are not very different whether changing direction, height or position, which shows that our APER-DDQN decision-making model has good generalization ability and can realize accurate airdrop in different environments. Specifically, when only changing the direction of throwing, the result changes little. Although the average reward decreases, the success rate is the same. When increasing the airdrop height on the basis of changing direction, the average reward increased by 0.1, but the success rate decreased by 0.04. This shows that although the number of successful hits has decreased in the test, there are more throws that get higher rewards, that is, more accurate. The average reward and success rate decreased when changing locations for airdrops, possibly due to airdrop errors caused by limited vision in some decisions. However, from the results, the average reward did not drop much, indicating that the performance of APER-DDQN is still stable enough.

### V. CONCLUSION AND FUTURE WORK

In order to solve the problem of accurate airdrop of UAV, we propose an adaptive priority experience replay DDQN (APER-DDQN) algorithm, which adds the priority experience replay mechanism, adaptive discount rate and learning rate on the basis of DDQN. Our method applies DRL to solve the airdrop problem for the first time, and improves the traditional DRL method. In order to better simulate the airdrop environment, we built a simulated airdrop experimental platform, and obtained the experimental samples and data through the experimental platform. The experimental results demonstrate that our APER-DDQN has superior performance than DDQN and DQN, and can achieve good results in solving the problem of accurate airdrop. Specifically, our APER-DDQN has a faster convergence speed, higher reward and more stable performance. In the airdrop test, the average reward of APER-DDQN is 7.07-7.63, the success rate is 0.83-0.9, while the average reward of DDQN is 5.9-6.3, the success rate is 0.67-0.77, the average reward of DQN is 4.9-5.43, the success rate is 0.53-0.63, the average reward for decision-making based on human experience is 4.1-4.57, and the success rate is 0.43-0.5. In future work, we consider increasing the complexity and diversity of the airdrop environment, further improving the performance of DRL decision-making models, and deploying the trained algorithms on UAV to solve practical problems.

### ACKNOWLEDGMENT

The authors would like to thank the Editor-in-Chief, the Associate Editor, and Reviewers for their insightful comments and suggestions.

### REFERENCES

[1] H. Menouar, I. Guvenc, K. Akkaya, A. S. Uluagac, A. Kadri, and A. Tuncer, “UA V-enabled intelligent transportation systems for the smart city: Applications and challenges,” IEEE Commun. Mag., vol. 55, no. 3, pp. 22–28, Mar. 2017.

[2] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, “Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments,” IEEE Access, vol. 9, pp. 24884–24900, 2021.

[3] Z. Guanlin, “Research on calculation and prediction of unpowered airdrop drop point,” M.S. thesis, College Astronaut., Nanjing Univ. Aeronaut. Astronaut., Nanjing, China, 2019.

[4] Q. Liu, J. W. Zhai, S. Zhong, Z. C. Zhang, Q. Zhou, and P. Zhang, “A deep recurrent q-network based on visual attention mechanism,” J. Comput. vol. 40, no. 6, pp. 1353–1366, 2017.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” Nature, vol. 518, pp. 529–533, 2015.

[6] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” in Proc. 30th Assoc. Adv. Artif. Intell. Conf. Artif. Intell. (AAAI), 2016, pp. 2094–2100.

[7] J. Koh, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” Int. J. Robot. Res., vol. 32, no. 11, pp. 1238–1274, 2013.

[8] F. Qiming, L. Quan, W. Hui, X. Fei, Y. Jun, and L. Jiao, “An off-policy Q(λ) algorithm based on linear function approximation,” Chin. J. Comput., vol. 37, no. 3, pp. 677–686, 2014.

[9] G. Yang, Z. Ruyi, W. Hao, and C. Zhixin, “Research on reinforcement learning algorithm of average reward,” J. Comput. Sci., vol. 518, pp. 1372–1378, 2007.

[10] G. Tesauru, “TD-Gammon, a self-teaching backgammon program, achieves master-level play,” Neural Comput., vol. 6, no. 2, pp. 215–219, 1994.

[11] E. Ipek, O. Mutlu, J. F. Martinez, and R. Caruana, “Self-optimizing memory controllers: A reinforcement learning approach,” ACM SIGARCH Comput. Archit. News., vol. 36, no. 3, pp. 39–50, Jun. 2008.

[12] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep reinforcement learning framework for autonomous driving,” Electron. Imag., vol. 2017, no. 19, pp. 70–76, 2017.
Y. Ouyang et al.: APER-DDQN: UAV Precise Airdrop Method Based on Deep Reinforcement Learning

[13] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA, USA: MIT Press, 2018.
[14] Y. Lin, M. Wang, X. Zhou, G. Ding, and S. Mao, “Dynamic spectrum interaction of UAV flight formation communication with priority: A deep reinforcement learning approach,” IEEE Trans. Cogn. Commun. Netw., vol. 6, no. 3, pp. 892–903, Sep. 2020.
[15] S. Lange and M. Riedmiller, “Deep auto-encoder neural networks in reinforcement learning,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2010, pp. 1–8.
[16] S. Lange, M. Riedmiller, and A. Voigtlander, “Autonomous reinforcement learning on raw visual input data in a real world application,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jun. 2012, pp. 1–8.
[17] C. J. C. H. Watkins, “Learning from delayed rewards,” Robot. Auton. Syst., vol. 15, no. 4, pp. 233–235, 1989.
[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013, arXiv:1312.5602.
[19] H. Hasselt, “Double Q-learning,” in Proc. Adv. Neural Inf. Process. Syst., vol. 23, 2010, pp. 1–9.
[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” 2015, arXiv:1511.05952.
[21] V. François-Lavet, R. Fonteneau, and D. Ernst, “How to discount deep reinforcement learning: Towards new dynamic strategies,” 2015, arXiv:1512.02011.
[22] M. J. Hausknecht and P. Stone, “Deep recurrent Q-learning for partially observable MDPs,” in Proc. AAAI Fall Symp., 2015, pp. 1–9.
[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2015, arXiv:1509.02971.
[24] V. Mnih, “Asynchronous methods for deep reinforcement learning,” in Proc. ICMRL, 2016, pp. 1928–1937.
[25] Z. Zhigang, Principles of Integrated Fire Control in Aviation. Beijing, China: National Defense Industry Press, 2008.
[26] X. Gao, Q. Zhang, and Q. Tang, “Parachute dynamics and perturbation analysis of precision airdrop system,” Chin. J. Aeronaut., vol. 29, no. 3, pp. 596–607, Jun. 2016.
[27] J. Berland, W. Gargano, B. Bagdonovich, and J. Barber, “Autonomous precision delivery of 42,000 pounds (19,000 kg) under one parachute,” in Proc. 20th AIAA Aerodyn. Decelerator Syst. Technol. Conf. Seminar, May 2009, p. 2928.
[28] J. Yin, J. Gao, Y. Liu, and W. Zhang, “Moving base initial alignment method of airborne load SINS in precise airdrop system,” Comput. Simul., vol. 33, no. 12, pp. 69–72 and 184, 2016.
[29] X. Zhaojun and L. Changxi, “Modeling and simulation of 9-DOF accurate airdrop system,” J. Huaqiao Univ. Sci. Technol., Natural Sci., vol. 38, no. 6, p. 4, 2010.
[30] M. Mingze, “Simplified airdrop trajectory simulation based on MATLAB/SIMULINK,” in Proc. IEEE Chin. Guid., Navigat. Control Conf. (GCNCC), 2016, pp. 2678–2681.
[31] T. Yang, L. Zhan, W. Li, J. Wu, and A. Y. Zomaya, “Towards healthy and cost-effective indoor environment management in smart homes: A deep reinforcement learning approach,” Appl. Energy, vol. 300, Oct. 2021, Art. no. 117335.
[32] H. Tan, H. Zhang, J. Peng, Z. Jiang, and Y. Wu, “Energy management of hybrid electric bus based on deep reinforcement learning in continuous state and action space,” Energy Convers. Manage., vol. 195, pp. 548–560, Sep. 2019.
[33] Y. Fu, C. Li, F. R. Yu, T. H. Luan, and Y. Zhang, “A decision-making strategy for vehicle autonomous braking in emergency via deep reinforcement learning,” IEEE Trans. Veh. Technol., vol. 69, no. 6, pp. 5876–5888, Jun. 2020.
[34] L.-J. Lin, Reinforcement Learning for Robots Using Neural Networks. Pittsburgh, PA, USA: Carnegie Mellon Univ., 1992.
[35] L. Xi, L. Yu, Y. Xu, S. Wang, and X. Chen, “A novel multi-agent DQN-AD method-based distributed strategy for automatic generation control of integrated energy systems,” IEEE Trans. Sustain. Energy, vol. 11, no. 4, pp. 2417–2426, Oct. 2020.
[36] C. Zheng, W. Li, W. Li, K. Xu, L. Peng, and S. W. Cha, “A deep reinforcement learning-based energy management strategy for fuel cell hybrid buses,” Int. J. Precis. Eng. Manuf.-Green Technol., vol. 9, pp. 885–897, Dec. 2021.
[37] W.-X. Liu, J. Lu, J. Cai, Y. Zhu, S. Ling, and Q. Chen, “DRL-P4Link: Deep reinforcement learning with private link approach for mix-flow scheduling in software-defined data-center networks,” IEEE Trans. Netw. Service Manage., early access, Nov. 16, 2021, doi: 10.1109/TNSM.2021.3126267.

YAN OUYANG was born in Hengyang, Hunan, China, in 1998. He is currently pursuing the master’s degree in mechanical engineering with Army Engineering University. His research interests include deep learning, reinforcement learning, and computer vision.

XINQING WANG received the Ph.D. degree (Hons.). He is currently a Professor and a Ph.D. Tutor at Army Engineering University, China. His main research interests include electromechanical control, intelligent signal processing, and computer vision.

RUIZHE HU was born in Xiangyang, Hubei, China, in 1999. He is currently pursuing the master’s degree in mechanical engineering with Army Engineering University. His research interests include deep learning, adversarial examples, and object detection.

HONGHUI XU was born in Shantou, Guangdong, China, in 1995. He is currently pursuing the master’s degree in mechanical engineering with the College of Field Engineering, Army Engineering University. His research interests include machine learning and computer vision, especially object detection.

* * *