Journal of Systems Engineering and Electronics
Vol. 35, No. 3, June 2024, pp.644 – 665

UAV maneuvering decision-making algorithm based on deep reinforcement learning under the guidance of expert experience

ZHAN Guang[1], ZHANG Kun[1,2,*], LI Ke[1], and PIAO Haiyin[1]

1. School of Electronics and Information, Northwestern Polytechnical University, Xi'an 710072, China;
2. Science and Technology on Electro-Optic Control Laboratory, Luoyang 471009, China

Abstract: Autonomous umanned aerial vehicle (UAV) manipulation is necessary for the defense department to execute tactical missions given by commanders in the future unmanned battlefield. A large amount of research has been devoted to improving the autonomous decision-making ability of UAV in an interactive environment, where finding the optimal maneuvering decision-making policy became one of the key issues for enabling the intelligence of UAV. In this paper, we propose a maneuvering decision-making algorithm for autonomous air-delivery based on deep reinforcement learning under the guidance of expert experience. Specifically, we refine the guidance towards area and guidance towards specific point tasks for the air-delivery process based on the traditional air-to-surface fire control methods. Moreover, we construct the UAV maneuvering decision-making model based on Markov decision processes (MDPs). Specifically, we present a reward shaping method for the guidance towards area and guidance towards specific point tasks using potential-based function and expert-guided advice. The proposed algorithm could accelerate the convergence of the maneuvering decision-making policy and increase the stability of the policy in terms of the output during the later stage of training process. The effectiveness of the proposed maneuvering decision-making policy is illustrated by the curves of training parameters and extensive experimental results for testing the trained policy.

Keywords: unmanned aerial vehicle (UAV), maneuvering decision-making, autonomous air-delivery, deep reinforcement learning, reward shaping, expert experience.

DOI: 10.23919/JSEE.2024.000022

1. Introduction

With rapid development, unmanned aerial vehicles (UAVs) have become important roles in various engineering fields in recent years. UAVs have been used to assist or replace human to execute dirty, boring, and difficult missions due to its low cost, high mobility, and unmanned feature [1]. Therefore, UAVs are widely used in surveillance, searching, tracking, and other missions [2]. Thus, how to improve the autonomy of UAVs while performing some tasks to avoid risking human lives has become the research focus in various fields. For instance, some people used UAVs to carry out delivery of relief supplies, extinguishing, and so on [3–5]. Consequently, it has become one of the key issues for engineering applications to improve the autonomous flight capability of UAV [6–9].

Nowadays, UAVs are mainly used to execute tasks which could be conducted automatically instead of manually, such as target tracking, long-distance delivery, patrol and so on. One of the important technical issues from these tasks is finding an optimal path from start to end point and designing a controller to manipulate it following the path. The optimal path could be found by path planning algorithms [10], such as visibility graph [11], randomly sampling search algorithms including rapidly-exploring random tree [12], probabilistic roadmap [13], heuristic algorithms including A-Star [14], Sparse A-Star [15], and D-Star [16], and genetic algorithms [17]. Then, a controller could be designed to operate UAV following the planned path using various trajectory tracking algorithms [18]. However, there are some disadvantages in the solution mentioned above. For example, finding the optimal path relies on prior knowledge about the environment, but the data of terrain and obstacles is so difficult to obtain that the capability of environment modelling is limited [19,20]. Moreover, for dynamic environment with moving obstacles [21], the scheme designed above is not flexible enough to alter their control strategies immediately. A replan of paths has to be scheduled to adapt to the

Manuscript received June 14, 2022.
*Corresponding author.
This work was supported by the Key Research and Development Program of Shaanxi (2022GXLH-02-09), the Aeronautical Science Foundation of China (20200051053001), and the Natural Science Basic Research Program of Shaanxi (2020JM-147).

changes in the environment. Furthermore, because conventional algorithms need much more time to calculate the optimal path, it is difficult to apply them to solving real-time problems. Therefore, it is necessary to design an end-to-end algorithm which could be used to operate autonomous UAV flight in a dynamic environment without path planning and trajectory tracking.

A research highlight is inspired by AlphaGo developed by Google based on deep reinforcement learning (DRL) [22], which could play Atari games using an end-to-end decision-making algorithm called deep Q network (DQN) [23]. The performance of this algorithm reached human level after an extensive training, and it has attracted lots of researchers from various fields to study the applications of DRL in all kinds of engineering problems [8,9]. Meanwhile, the deep deterministic policy gradient (DDPG) [24] was proposed to solve the dimension explosion caused by the continuity of action space and state space. And the experience replay method that samples from experience buffer was constructed in these algorithms to allow agents to remember and learn from historical data. The uniform experience replay (UER) was used to form training set by taking samples from experience buffer. However, the UER does not fully exploit the diversity of historical data and affects the convergence rate of policy, even causes divergence after lots of training. Therefore, the prioritized experience replay (PER) [25] was proposed to improve the efficiency of learning from experiences. It uses the potential value of historical data to increase the convergence rate of policy network, because a priority model of each sample is designed to evaluate the profit of samples for training of policy network at current step.

In addition, it is important for training policy to design a reward function of target problem. Traditional formulation of reward function comes from original model of the problem [23,24], such as CartPole, Pendulum and other Atari games. Thus, how to construct an appropriate reward function is not the focus of most popular works, as they mainly focus on the improvements of algorithms instead of modelling. However, it is also essential for solving problems from specific research fields to define modified model when we attempt to apply DRL for a new problem. Traditionally, some researchers considered that the reward function involved in problem model is designed by human experience [26,27] so that the trained policy extremely relied on the capability of designer. Although some recently published work found effective policy, we could find that different formulations of reward function brought different training processes and trained policies.

In the present work, we aim to tackle the challenges mentioned above and focus on the UAV maneuvering decision-making algorithm for autonomous air-delivery including guidance towards area and guidance towards specific point tasks. The main works presented in this paper are summarized as follows:

(i) The UAV maneuvering decision-making model for air-delivery is built based on Markov decision processes (MDPs) [28,29]. Particularly, we refine the guidance towards area and guidance towards specific point tasks involved in the air-delivery problem. Meanwhile, we design the flight state space, the flight action space, and the reward functions of each task. Among the components of the model, we devote the traditional air-to-ground fire control theory to designing and constructing the UAV maneuvering decision-making model for air-delivery.

(ii) We propose the UAV maneuvering decision-Making algorithm for autonomous air-delivery based on DDPG with PER sampling (PER-DDPG) to optimize the maneuvering policy. Specially, we design the policy function by deep neural network and generate the training samples based on PER. Moreover, we present a construction method of reward function based on expert experience and domain knowledge.

(iii) It is proved that the proposed algorithm could improve the autonomy of UAV during the air-delivery process and the presented construction method of reward function is beneficial to the convergence of maneuvering policy, even improving the quality of policy's output.

This paper is organized as follows: Section 2 describes the background knowledge of all the methods used to design UAV maneuvering decision-making model and algorithm for air-delivery. Section 3 presents the details of experiments we design, comparison of training parameters under different sampling methods and reward functions. Section 4 shows the conclusion of our work and looks forward to the future of our research.

2. Methodology

With the rapid development of UAV technology, it has been used to execute various dangerous and repetitive missions, such as electric power inspection, crop protection, wildlife surveillance, traffic monitoring, and rescue operations. Demands for more advanced and simple UAV autonomous flight solution have emerged. As mentioned above, traditional solution of UAV guidance is that the algorithm first plans an optimal path and then UAV follows the path by the trajectory tracking method.

In this paper, we describe the process of UAV

autonomous air-delivery in detail and define the guidance towards area and guidance towards specific point tasks involved in the air-delivery problem. Then, we construct the UAV maneuvering decision-making model for air-delivery based on MDPs. Meanwhile, we present a construction method of reward function, and the expert experience and domain knowledge are given due consideration. On the other hand, we propose the UAV maneuvering decision-making algorithm based on DRL.

As shown in Fig. 1, we first construct the UAV maneuvering decision-making model for air-delivery consisting of the guidance towards area and guidance towards specific point tasks based on MDPs. Among this model, we design action space, state space and basic reward of each task which are used to demonstrate the characteristics of UAV autonomous flight during air-delivery. Moreover, we design and realize the UAV maneuvering decision-making algorithm including the UAV maneuvering decision-making policy based on neural network, and the policy network is optimized according to data sampled from historical experiences by PER. Meanwhile, we construct the shaping reward of each task to increase the convergence rate of policy network and to improve the quality of policy's output.

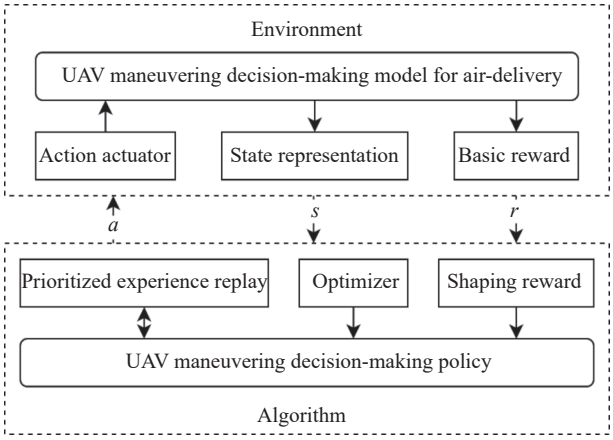


Fig. 1 Structure of UAV maneuvering decision-making for air-delivery based on DRL

2.1 UAV maneuvering decision-making model for air-delivery based on MDPs

At present, the most cyclical decision-making problems, called multi-stage decision-making, could be modeled by MDPs. Meanwhile, most of the researchers focusing on autonomous control and decision-making always describe problems and construct the problem model based on MDPs, including the UAV maneuvering decision-making model for air-delivery. As shown in Fig. 2, we use MDPs to design and construct the UAV maneuvering decision-making model for the guidance towards area and guidance towards specific point tasks included in air-delivery. We design and implement the simulator core consisting of UAV kinematic model, bomb model and target point. Particularly, we present the state space, action space and reward function of UAV maneuvering decision-making model for air-delivery in the light of MDPs.

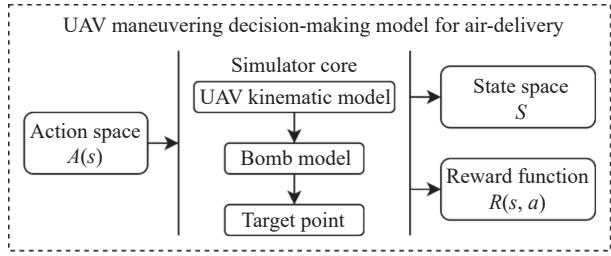


Fig. 2 Construction diagram of UAV maneuvering decision-making model for air-delivery based on MDPs

2.1.1 MDPs

During the process of performing air-delivery mission, UAV maneuvering decision-making could be regarded as a sequential decision processes. Moreover, the operator of UAV usually considers current information from environment while selecting optimal action. Therefore, we can consider this decision process Markovian and use MDPs to model the UAV maneuvering decision-making model for air-delivery.

The MDPs can be described by a tuple

$$\{T, S, A(s), P(\cdot|s,a), R(s,a)\}$$

where T represents the decision episode, S represents the state space, $A(s)$ represents the action space, and the transition probability $P(\cdot|s,a)$ represents the probability distribution of the environment's state at the next moment when the action $a \in A(s)$ is executed in the environment in the state $s \in S$. The reward function $R(s,a)$ represents the benefit that agent gets when $a \in A(s)$ is taken in the state $s \in S$. Then, we can make a complete mathematical description of the sequence decision problem based on MDPs.

As shown in Fig. 3, MDPs could be described as follows: when the state of environment is initialized by s_0, the agent chooses the action $a_0 = \pi(s_0)$, where $\pi(s)$ represents the policy of agent, and the state of environment will be updated to s_1 according to $P(s_1|s_0,a_0)$. Meanwhile, the environment also returns the reward signal $r_0 = R(s_0,a_0)$ to agent. This process mentioned above will end until the state of environment becomes termination state.

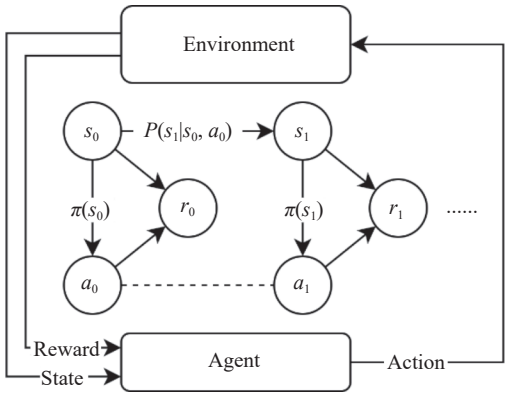


Fig. 3 Structure of finite MDPs and traditional scheme of solving MDPs

During the process of interactions between environment and agent, a sequence of rewards $(r_0, r_1, \cdots)$ is produced. Among this process, agent is stimulated by external rewards, and the policy of agent will converge when the expectation of future rewards about policy is the maximum. Therefore, the utility function (in the state $s \in S$, the expected rewards obtained by adopting the policy $\pi(s)$) is $v(s,\pi)$. When the current policy is the optimal, it should be satisfied as follows:

$$v(s) = \sup_{\pi} v(s,\pi), \quad s \in S. \tag{1}$$

Based on the characteristics of the UAV maneuvering decision-making problem for air-delivery, we use the infinite stage discount model as the utility function

$$v(s,\pi) = \sum_{t=0}^{\infty} \gamma^t \mathrm{E}_{\pi}^{s}[R(s_t, a_t)], \quad s \in S. \tag{2}$$

In the above formula, $\gamma \in (0,1)$ is the future reward discount factor, and $\mathrm{E}[\cdot]$ represents mathematical expectation. Then, the optimal policy under the discount model can be obtained by solving (2).

In the following, we will first demonstrate the problem definition among air-delivery missions. Then, state space S, action space $A(s)$, transition probability model $P(\cdot|s,a)$ and reward function $R(s,a)$ of each task will be designed.

2.1.2 Definition of guidance towards area and guidance towards specific point tasks involved in air-delivery mission

Before we start running a reinforcement learning based algorithm, we should construct a simulation model of problems to be solved. Thus, we refine two tasks involved in air-delivery, which are the guidance towards area task and guidance towards specific point task. As shown in Fig. 2, we should first construct UAV kinematic model and bomb model. In this paper, we adopt a dynamic model describing air-delivery mission based on 3-degree of freedom (3-DoF) kinematic model of UAV [6]. On the other hand, we also design a 3-DoF kinematic model of bomb [30,31] to calculate the external ballistics parameters of uncontrolled bomb. When the position and attitude of UAV are confirmed at t, we can obtain the state of UAV at $t+1$ by solving the model we designed. Therefore, we consider the transition probability of UAV maneuvering decision-making model to be $p(\cdot|s,a) = 1$, which belongs to the deterministic model.

Based on the 3-DoF kinematic model of UAV, the flight state is defined as $(x, z, v, \psi_{\mathrm{UAV}})$, where (x, z) is the horizontal coordinate of the UAV in the geographical coordinate system, and ψ_{UAV} is the azimuth angle of the UAV flight path. On the other hand, the steering overload of UAV is defined as $N_z \in \left[-N_z^{\max}, N_z^{\max}\right]$, and $N_z^{\max}$ represents the max normal acceleration of UAV in the body coordinate system. During the simulation process, the algorithm outputs the current optimal maneuvering control N_z and the next state $(x, z, v, \psi_{\mathrm{UAV}})'$ of UAV is calculated with the current state $(x, z, v, \psi_{\mathrm{UAV}})$ according to the flight simulation model of UAV. Meanwhile, we could obtain the impact point of the bomb released from current position of UAV based on the external ballistics parameters of uncontrolled bomb.

In this paper, the guidance towards area task and guidance towards specific point task are defined and described, considering the task load type and launching mode.

(i) Guidance towards area task

When the UAV flies with controlled bomb, or other steerable loads, the guidance target of UAV is usually a broad area. Thus, we consider the target of UAV to be an area including mission point, while using the controlled bomb.

Fig. 4 is a vector diagram of guidance towards area task.

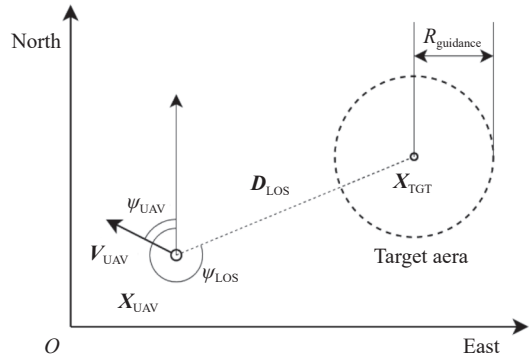


Fig. 4 Diagram of guidance towards area task involved in air-delivery mission

$\boldsymbol{X}_{\text{UAV}}$ and $\boldsymbol{V}_{\text{UAV}}$ represent the current position of UAV and the current velocity of UAV respectively. $\boldsymbol{X}_{\text{TGT}}$ represents the target point. ψ_{LOS} indicates the azimuth of line of sight (LOS) and $\boldsymbol{D}_{\text{LOS}}$ represents the distance vector between UAV and target point, which could be defined by

$$\boldsymbol{D}_{\text{LOS}} = \boldsymbol{X}_{\text{TGT}} - \boldsymbol{X}_{\text{UAV}}. \tag{3}$$

Thereby, we could define the successful termination condition of guidance towards area task

$$\|\boldsymbol{D}_{\text{LOS}}\|_2 \leqslant R_{\text{guidance}}. \tag{4}$$

In the formula above, $\|\cdot\|_2$ represents the 2-norm of vector. Finally, when UAV enters the target area, it could launch the controlled bomb.

(ii) Guidance towards specific point task

When the mission load of UAV is uncontrolled bomb, the target of the task is a specific point instead of an area. As shown in Fig. 5, if UAV flies with uncontrolled bomb, it will perform guidance towards specific point task and adjust $\boldsymbol{D}_{\text{LOS}}$ and $\delta_{\psi_{\text{LOS}}}$ when satisfying the release condition of bomb.

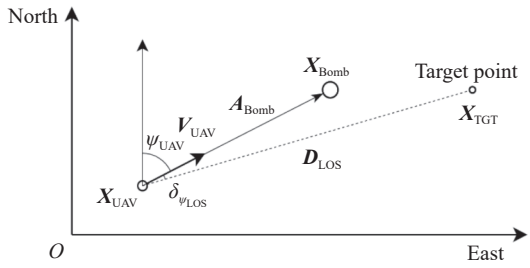


Fig. 5 Diagram of guidance towards specific point task involved in air-delivery mission

In Fig. 5, $\delta_{\psi_{\text{LOS}}}$ represents the relative azimuth between LOS and the nose direction of UAV, which could be calculated by

$$\delta_{\psi_{\text{LOS}}} = \psi_{\text{UAV}} - \psi_{\text{LOS}}. \tag{5}$$

$\boldsymbol{A}_{\text{Bomb}}$ represents the range vector of bomb. Thereby, we could obtain the impact point of bomb $\boldsymbol{X}_{\text{Bomb}}$ by

$$\boldsymbol{X}_{\text{Bomb}} = \boldsymbol{X}_{\text{UAV}} + \boldsymbol{A}_{\text{Bomb}}. \tag{6}$$

Moreover, we could define the successful termination condition of guidance towards specific point task

$$\begin{cases} \left|\delta_{\psi_{\text{LOS}}}\right| \leqslant \delta_{\psi} \\ \|\boldsymbol{X}_{\text{TGT}} - \boldsymbol{X}_{\text{Bomb}}\|_2 \leqslant \delta_A \end{cases} \tag{7}$$

where δ_{ψ} and δ_A indicate the minimum of azimuth deviation and distance error respectively.

2.1.3 State space, action space, and basic reward of each task

As mentioned above, we present the UAV maneuvering decision-making model for air-delivery based on MDPs. Therefore, we should design state space, action space and reward function of each task refined in air-delivery based on the problems' definition described above.

(i) State space of guidance towards area task

Considering that the purpose of guidance towards area task, we define the state space of guidance towards area task as

$$S_{\text{guidance}} = \{D_{\text{LOS}}, \delta_{\psi_{\text{LOS}}}, v_{\text{UAV}}, H_{\text{UAV}}\} \tag{8}$$

where $D_{\text{LOS}} \in [D_{\text{LOS}}^{\min}, D_{\text{LOS}}^{\max}]$ represents the distance between the UAV and the target point, v_{UAV} and H_{UAV} represents the speed and height of the UAV.

(ii) State space of guidance towards specific point task

For the guidance towards specific point task, we can define its state space as

$$S_{\text{aim}} = \{D_{\text{LOS}}, \delta_{\psi_{\text{LOS}}}, v_{\text{UAV}}, H_{\text{UAV}}, A_{\text{Bomb}}\} \tag{9}$$

where A_{Bomb} represents the horizontal range of bomb under current situation of environment.

(iii) Action space of each task

Based on the flight simulation model of UAV we construct, we can establish the action space as

$$A(s) = \{N_z\}. \tag{10}$$

In the guidance towards specific point task, the action space is defined as (11) similarly.

(iv) Reward function of each task

Moreover, considering the successful termination condition of each task, we define the reward function as

$$R(s,a) = \begin{cases} 1.0, & \text{Successful termination} \\ 0.0, & \text{Failed} \end{cases}. \tag{11}$$

The formula above shows that if UAV's situation satisfies the successful termination condition of each task, $R(s,a)$ will return 1.0, otherwise 0.0. Thereby, this reward will encourage agent to find policy that maximizes the expectation of future rewards.

2.2 UAV maneuvering decision-making algorithm for air-delivery based on PER-DDPG

After constructing simulation environments, we could design the corresponding algorithm to solve the problem. In this paper, we propose the UAV maneuvering decision-making algorithm for air-delivery mission based on PER-DDPG [6] with expert experience and domain knowledge. It is composed of the PER to generate training samples, the policy including actor network and critic network, and shaping reward to improve the quality of policy's output and increase the convergence rate of policy, as shown in Fig. 6. Particularly, we introduce expert

experience and domain knowledge to design the shaping reward to achieve the improved performance of the proposed algorithm.

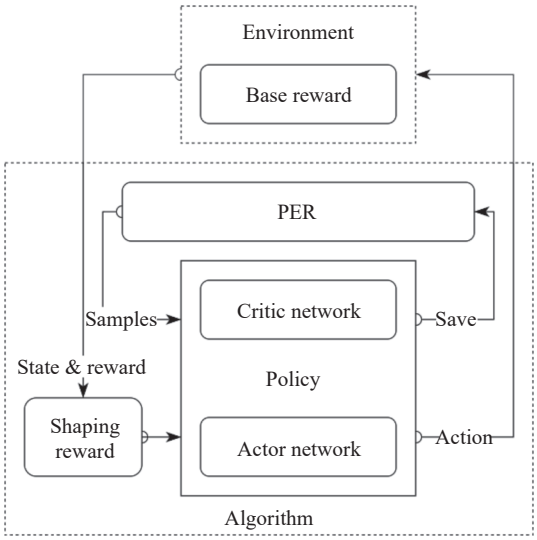


Fig. 6 Running process of UAV maneuvering decision-making algorithm for air-delivery

2.2.1 Framework of PER-DDPG

The PER-DDPG is a model-free, off-policy and DRL-based algorithm designed by actor-critic architecture. Problems belong to MDPs have continuous state space and action space and could be solved effectively. Meanwhile, because DDPG does not consider the diversity of data and does not fully utilize historical experience, policy converged in DDPG exhibit low convergence rate and poor stability. Therefore, the PER is used to generate training data, which can improve the utilization of the potential value of historical data, thereby increasing convergence rate and enhancing the stability of trained policy. PER-DDPG has been verified in autonomous airdrop task, showing the high performance than original DDPG.

The framework of PER-DDPG is shown in Fig. 7, which is composed of the evaluation networks, target networks, the PER and other methods. At each decision-making step, the evaluation actor network outputs action with noise for exploring according to state. Then, the current state, action, reward, and next state are packaged and stored in historical transitions buffer D. During the process of storing transition, sample binds with probability are used for PER sampling. When we are training networks' parameters, the training data are sampled from D by PER, and the temporal difference error (TD-error) of each data are accumulated for updating the evaluation of networks' parameters with importance sampling (IS) weights and used to update the priority of data after training. Finally, the parameters of target networks are also updated smoothly.

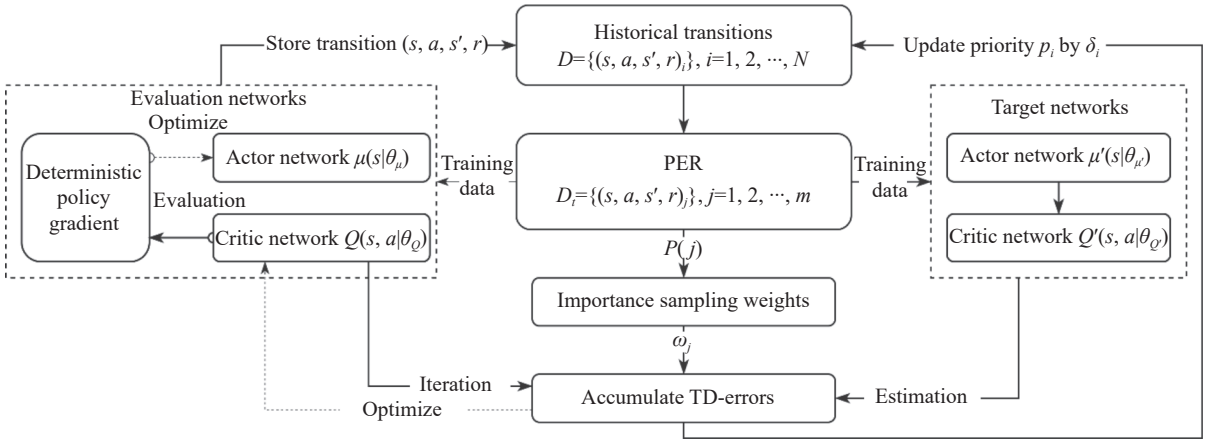


Fig. 7 Diagram of PER-DDPG's framework

Specifically, at each moment, the policy gives action by

$$a_t = \mu(s_t|\theta_\mu) \tag{12}$$

where $s_t \in S$ is current state and a_t is the output of actor network $\mu(s|\theta_\mu)$. During the training process, the critic network $Q(s,a|\theta_Q)$ evaluates current action given by actor network and this value is used for the basis of updating $\mu(s|\theta_\mu)$.

2.2.2 UAV maneuvering decision-making policy based on neural network

As mentioned above, DDPG is a kind of DRL algorithm constructed by the actor-critic framework. During the training process, the actor network $\mu(s|\theta_\mu)$ outputs action $a \in A(s)$ according to the state $s \in S$ of environment.

Meanwhile, TD-error is used to optimize the parameters of critic network $Q(s,a|\theta_Q)$. Similarly, the parameters of actor network $\mu(s|\theta_\mu)$ are optimized according to $\max_{\theta_\mu} Q(s,a|\theta_Q)$. Therefore, we must design the structure of actor and critic networks respectively based on DL.

(i) Actor network

The actor network $\mu(s|\theta_\mu)$ is mainly used to output action in real-time decision according to state. The input of network is the environment state $s \in S$, and its output is policy's action $a \in A(s)$. The structure of the designed actor network is shown in Fig. 8.

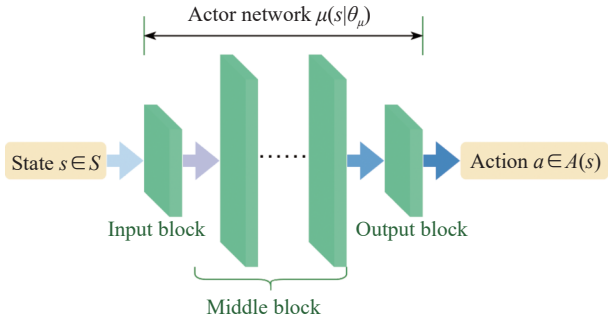


Fig. 8 Structure of actor network

(ii) Critic network

The critic network $Q(s,a|\theta_Q)$ is used to evaluate the advantage of current action $a \in A(s)$ output by $\mu(s|\theta_\mu)$. The input of $Q(s,a|\theta_Q)$ is the combination of state and action $[s,a]$, and the output of $Q(s,a|\theta_Q)$ is the state-action value function $Q(s,a)$. The structure of the designed critic network is shown in Fig. 9.

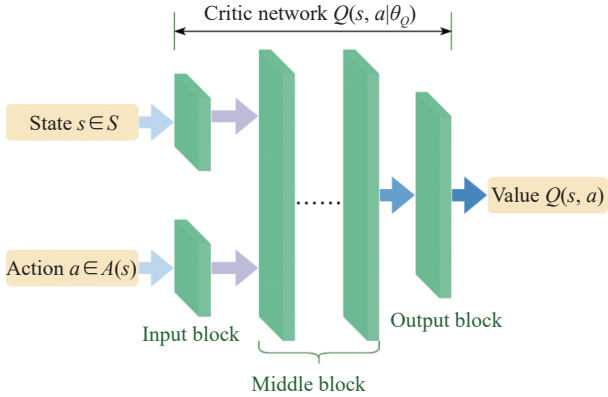


Fig. 9 Structure of critic network

In addition, before running the networks, the input value should be normalized for eliminating the influence of data's physical meaning. Moreover, the structure of target networks $Q'(s,a|\theta_{Q'})$ and $\mu'(s|\theta_{\mu'})$ is similar to $Q(s,a|\theta_Q)$ and $\mu(s|\theta_\mu)$, and only the method of parameters updating is different.

2.2.3 Shaping reward function based on expert experience and domain knowledge

Although the algorithm we propose could learn an optimal policy according to the reward function shown in (12), there is a serious challenge that could influence the convergence rate of policy, because the rewards environment returned are too sparse to learn useful experience such as those transitions whose reward is not zero.

Therefore, some researchers proposed a technique called reward shaping (RS) [32], which leverages the expert knowledge to reconstruct the reward model of the target domain to improve the agent's policy learning. More specifically, in addition to reward from environment, RS provides a shaping function $F: S \times S \times A \to \mathbf{R}$ to render auxiliary rewards. Intuitively, RS will assign higher rewards to more beneficial transitions, which could guide the agent to find the optimal policy quickly. As a result, the agent will learn its policy by the newly shaped rewards $R' = R + F$, which means that RS has altered the original reward with a different shaping reward function

$$M = \{T, S, A, P, R\} \to M' = \{T, S, A, P, R'\}. \tag{13}$$

In the formula above, M is the original problem modelled by MDPs, similar to (1), and M' represents the improved model with shaping reward function. Along the line of RS, the potential-based RS (PBRS) [33] is one of the most classical approaches. PBRS gives the algorithm an external signal for learning the optimal policy more quickly than before by adding a new reward shaping function $F(s,a,s')$, formed by the difference between two potential functions $\Phi(\cdot)$

$$F(s,a,s') = \gamma\Phi(s') - \Phi(s) \tag{14}$$

where $\Phi(\cdot)$ comes from the knowledge of expertise and evaluates the quality of a given state. There is a cycle dilemma existing in MDPs, which could be addressed by the potential difference. In the cycle dilemma, an agent can get positive rewards by following a sequence of states which form a cycle $\{s_1, s_2, \cdots, s_n, s_1\}$ with

$$F(s_1,a_1,s_2) + F(s_2,a_2,s_3) + \cdots + F(s_n,a_n,s_1) > 0.$$

Fortunately, PBRS could avoid this issue by making any state cycle meaningless, with $\sum_{i=1}^{n-1} F(s_i,a_i,s_{i+1}) \leqslant -F(s_n,a_n,s_1) \leqslant 0$. It has been proved that, without further restrictions on the original MDPs or the shaping function, PBRS is sufficient and necessary to preserve the policy invariance. Moreover, the optimal $Q(s,a)$ in the original model and the improved model are related by

$$Q_{M'}(s,a) = Q_M(s,a) - \Phi(s). \tag{15}$$

Thereby, if we construct a function over both the state and the action to form the potential function $\Phi(s,a)$, PBRS will be extended to a novel approach, called the potential based state-action advice (PBA) [34], which could evaluate how beneficial an action is to take from state

$$F(s,a,s',a') = \gamma\Phi(s',a') - \Phi(s,a). \tag{16}$$

Similar to (15), the optimal $Q(s,a)$ in both MDPs model are connected by

$$Q_{M'}(s,a) = Q_M(s,a) - \Phi(s,a). \tag{17}$$

Similarly, PBA is also sufficient and necessary to preserve the policy invariance. Therefore, once the optimal policy in M' is learned, the optimal policy in M can be recovered by

$$\mu_M(s) = \arg\max_{a\in A(s)}[Q_{M'} - \Phi(s,a)]. \tag{18}$$

Thus, we propose a construction approach to design the shaping function of air-delivery mission based on PBRS and PBA, introducing expert experience and domain knowledge, as shown in Fig. 10.

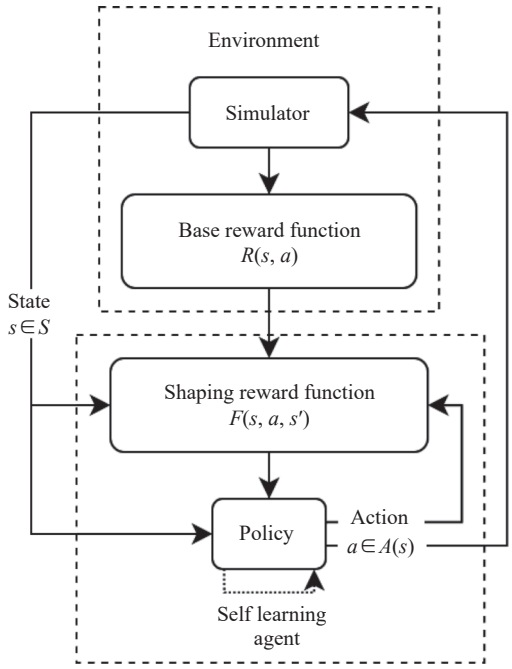


Fig. 10 Construction diagram of UAV maneuvering decision-making model for air-delivery

(i) Shaping function of guidance towards area task

When the UAV is performing the guidance towards area task, the distance between current position and target point and the relative azimuth between LOS and the nose direction of UAV are the main influence factors. Therefore, we can construct the shaping function as

$$F_{\text{guidance}}(s,a,s') = \gamma[\Phi_d(s') + \Phi_\psi(s')] - [\Phi_d(s) + \Phi_\psi(s)] \tag{19}$$

where $\Phi_d(s)$ represents the distance potential function, and $\Phi_\psi(s)$ represents the angle potential function. Considering the normalization of different influence factors, we can design $\Phi_d(s)$ and $\Phi_\psi(s)$ as

$$\Phi_d(s) = \frac{D_{\text{LOS}}^{\max} - D_{\text{LOS}}}{D_{\text{LOS}}^{\max} - D_{\text{LOS}}^{\min}}, \tag{20}$$

$$\Phi_\psi(s) = \frac{\pi - \delta_{\psi_{\text{LOS}}}}{\pi}. \tag{21}$$

According to the shaping function designed above, we could get optimal policy more quickly than sparse reward model. More specifically, we can introduce expert experience to navigate agent to reach optimal state faster

$$F_{\text{guidance}}(s,a,s',a') = F_{\text{guidance}}(s,a,s') + R^\dagger \tag{22}$$

where $R^\dagger$ represents the advice reward coming from the expert experience, which can be defined by

$$R^\dagger = \begin{cases} 1.0, & \Phi(s') \geqslant \Phi(s) \\ 0.0, & \text{otherwise} \end{cases} \tag{23}$$

which shows that if agent moves from the state with lower potential, the agent will receive a positive reward signal.

(ii) Shaping function of guidance towards specific point task

Similar to the shaping function of guidance towards area task, the shaping function of guidance towards specific point task can be modelled as

$$F_{\text{aim}}(s,a,s') = \gamma[\Phi_a(s') + \Phi_\psi(s')] - [\Phi_a(s) + \Phi_\psi(s)] \tag{24}$$

where $\Phi_a(s)$ represents the impact deviation potential function different to guidance towards area task, and it can be defined by

$$\Phi_a(s) = \exp\left(-\frac{D_{\text{Impact}} - D_{\text{Impact}}^{\min}}{D_{\text{Impact}}^{\max} - D_{\text{Impact}}^{\min}}\right) \tag{25}$$

where $D_{\text{Impact}} \in \left[D_{\text{Impact}}^{\min}, D_{\text{Impact}}^{\max}\right]$ represents the impact deviation, which can be calculated by

$$D_{\text{Impact}} = \|\boldsymbol{X}_{\text{TGT}} - \boldsymbol{X}_{\text{Bomb}}\|_2. \tag{26}$$

Furthermore, because of the accuracy of guidance towards specific point task, the shaping function with expert experience can be modelled by

$$F_{\text{aim}}(s,a,s',a') = F_{\text{aim}}(s,a,s') + R^\dagger(s,a,s',a'). \tag{27}$$

In the formula defined above, $R^\dagger(s,a,s',a')$ represents the shaping reward introducing expert experience, which can be calculated by

$$R^{\dagger}(s,a,s',a')=\exp\left[-\frac{\left|N_z-\left|\delta_{\psi_{\rm LOS}}\right|\cdot N_z^{\max}/\pi\right|}{2N_z^{\max}}\right] \tag{28}$$

which means that if the deviation between the nose direction of UAV and the azimuth of LOS is less than the last decision-making step, UAV will get a positive reward proportional to the difference between the action given by agent and the advice from expert.

2.2.4 Training set sampling method based on PER

During the training process, the PER [25] is used to sample training data from D in order to fully utilize the diversity of experiences. Usually, the training dataset D_t is constructed by selecting a batch of transitions from D uniformly, that means the probability $P(i)$ of each sample selected in D is equal. On the contrary, $P(i)$ of PER is not same, as defined in

$$P(i)=\frac{p_i^{\alpha}}{\sum\limits_k p_k^{\alpha}} \tag{29}$$

where p_i is the priority of the ith sample in D, and α is a hyperparameter. When $\alpha=0$, it is pure UER. p_i is defined based on TD-error, as shown in

$$p_i=|\delta_i|+\varepsilon \tag{30}$$

where δ_i is TD-error of the ith sample in D. In addition, a minimum ε is introduced to prevent p_i from being zero.

PER improves the availability of experiences, but the distribution bias of transitions sampled by PER occurs compared with UER, and this issue also reduces the diversity of training data. Therefore, IS weights are used to correct the distribution bias of training data caused by PER. The IS weight ω_j can be calculated by

$$\omega_j=\left(\frac{1}{N}\cdot\frac{1}{P(j)}\right)^{\beta} \tag{31}$$

where N is the size of D. When $\beta=1$, the distribution bias of training set is fully compensated. When we get δ_j, the updating target is $\omega_j\cdot\delta_j$, which is used to replace the target of DDPG.

In order to ensure stable convergence of $Q(s_j,a_j|\theta_Q)$, ω_j is normalized by $\dfrac{\omega_j}{\max\limits_i\omega_i}$. Thereby, the real IS weight ω_j can be defined as

$$\omega_j=\left(\frac{\min\limits_i P(i)}{P(j)}\right)^{\beta}. \tag{32}$$

At the same time, in the early stage of training, the distribution bias caused by PER is little. Therefore, we define an initial $\beta_0\in(0,1)$, which gradually increases to 1 with training going on.

2.2.5 Training procedure of UAV maneuvering decision-making algorithm

According to MDPs, the key issue of optimizing UAV maneuvering decision-making policy is to solve an optimization problem defined as

$$Q(s,a)=\mathrm{E}_{\pi}[v(s,\pi)] \tag{33}$$

where $v(s,\pi)$ is defined in (2). In this paper, double Q-learning is used to optimize $Q(s,a)$, as defined by

$$Q(s,a)=Q(s,a)+\sigma\left[r+\gamma Q'\left(s',\arg\max_a Q(s',a)\right)-Q(s,a)\right] \tag{34}$$

where $s\in S$ is the current state, $a\in A(s)$ is action, r is reward, $s'\in S$ is the next state, and $\sigma\in[0,1]$ is the learning rate. The loss function $L(\theta_Q)$ of critic network could be defined by

$$L(\theta_Q)=\mathrm{E}_{(s,a,r,s')_j}\left[\delta_j^{\,2}\right] \tag{35}$$

where δ_j is TD-error of the jth transition from D by PER. TD-error is defined as

$$\delta_j=y_j-Q(s_j,a_j|\theta_Q) \tag{36}$$

where y_j is the optimal objective of $Q(s_j,a_j|\theta_Q)$, $(s,a,r,s')_j$ represents the jth sample for training. Specifically, s_j and a_j are the current state and action in $(s,a,r,s')_j$ respectively. y_j could be calculated by

$$y_j=\begin{cases}r_j, & s_j \text{ satifies termination condition}\\ r_j+\gamma Q'(s'_j,\mu'(s'_j|\theta_{\mu'})|\theta_{Q'}), & \text{otherwise}\end{cases} \tag{37}$$

where r_j and s'_j are the current reward and next state in $(s,a,r,s')_j$. Thus, we can obtain the gradient of loss function $L(\theta_Q)$ as shown in

$$\nabla_{\theta_Q}L(\theta_Q)=\mathrm{E}_{(s,a,r,s')_j}\left[\delta_j\cdot\nabla_{\theta_Q}Q(s_j,a_j|\theta_Q)\right]. \tag{38}$$

Therefore, when we use PER to sample training data, the updating goal Δ of $Q(s_j,a_j|\theta_Q)$ could be calculated by

$$\Delta=\sum_j\omega_j\cdot\delta_j\cdot\nabla_{\theta_Q}Q(s_j,a_j|\theta_Q). \tag{39}$$

Meanwhile, we define the loss function $L(\theta_\mu)$ of actor network $\mu(s|\theta_\mu)$ as

$$L(\theta_\mu)=\mathrm{E}_s[Q(s,\mu(s|\theta_\mu)|\theta_Q)]. \tag{40}$$

Thereby, we can obtain the gradient of $L(\theta_\mu)$ according to the deterministic policy gradient theorem, as shown in

$$\nabla_{\theta_\mu}L(\theta_\mu)=\mathrm{E}_s\left[\nabla_{\theta_\mu}\mu(s|\theta_\mu)\nabla_a Q(s,a|\theta_Q)\big|_{a=\mu(s|\theta_\mu)}\right]. \tag{41}$$

In addition, considering stability of target networks' training, the parameters of $\mu'(s|\theta_{\mu'})$ and $Q'(s,a|\theta_{Q'})$ are updated by soft updating, like smooth updating, as shown in

$$\begin{cases}\theta_{Q'} = \tau\theta_Q + (1-\tau)\theta_{Q'} \\ \theta_{\mu'} = \tau\theta_\mu + (1-\tau)\theta_{\mu'}\end{cases}. \tag{42}$$

In the equation above, $\tau \in (0,1)$ is a hyperparameter involved in the soft updating. Moreover, in order to improve the explorationability of deterministic policy involved in algorithm, random noise is added into the action generated by $\mu(s|\theta_\mu)$, as shown in

$$a_t = \mu(s_t|\theta_\mu) + N(0,\sigma). \tag{43}$$

where $N(0,\sigma)$ represents the noise sampled by the normal distribution constructed by variance σ.

Finally, the training procedure of the UAV maneuvering decision-making algorithm for air-delivery is given in Algorithm 1.

Algorithm 1 UAV maneuvering decision-making algorithm for air-delivery

Input:

The hyperparameters of RL: policy's learning period K, historical buffer capacity N, soft updating parameter τ;

The hyperparameters of DL: size of minibatch k, actor networks' learning rate η_μ, critic networks' learning rate η_Q;

The hyperparameters of PER: availability exponent α, initial IS exponent β_0;

The hyperparameters of environment: maximum training episodes M, maximum steps per episode T

Output:

The evaluation networks: actor network $\mu(s|\theta_\mu)$ and critic network $Q(s,a|\theta_Q)$;

The target networks: actor network $\mu'(s|\theta_{\mu'})$ and critic network $Q'(s,a|\theta_{Q'})$

Run:

Initialize $Q(s,a|\theta_Q)$ and $\mu(s|\theta_\mu)$, and their target networks $Q'(s,a|\theta_{Q'})$ and $\mu'(s|\theta_{\mu'})$ using the parameters of evaluation networks

For m = 1 to M do

Reset environment and receive the initial state s_0

Calculate a_0 according to (43)

For t = 1 to T do

Calculate a_t by (43) according to current state s_t of environment, and receive the reward r_t and next state s_{t+1} from environment after executing action

Save current transition (s_t, a_t, r_t, s_{t+1}) into historical data buffer D

If t mod $K \equiv 0$ then

Clear the cumulative updating value Δ of $Q(s,a|\theta_Q)$

For j = 0 to k do

Sample data $j \sim P(j)$ according to (29)

Calculate IS weight ω_j according to (32)

Calculate TD-error δ_j of sampled transition according to formula and update its priority p_j according to formula

Accumulate Δ according to (39)

End for

Update the parameters of $Q(s,a|\theta_Q)$ according to Δ with η_Q

Update the parameters of $\mu(s|\theta_\mu)$ based on (41) with η_μ

Update the parameters of target networks $Q'(s,a|\theta_{Q'})$ and $\mu'(s|\theta_{\mu'})$ according to (42) with τ

End if

End for

End for

3. Results and analysis

Based on the model and algorithm we present above, experiments are conducted to prove the rationality of the model and verify the availability of the algorithm. In the following, we will explain settings of the simulation, details of the training process, results of Monte-Carlo (MC) test experiments, as well as their analysis.

3.1 Settings of simulation experiments

For the designed experiments, the mission area is restricted to 100 km × 100 km airspace and the height of UAV is bound to 500 – 5 000 m. For each simulation experiment of guidance towards area and guidance towards specific point tasks, the UAV's initial state is randomly generated, and the UAV might start from arbitrary position in mission area. In order to make simulation data as close to real world as possible, we define simulation step $T_s = 0.1$ s because operator of UAV always manipulates it every 0.5 – 1 s.

Moreover, because each element of state space in guidance towards area task and guidance towards specific point task has different physical units, all the dimensions of each vector before inputting into networks should be normalized. Details of parameters defined above are explained in Table 1. Thereby, we can normalize these parameters according to their range.

Table 1 Details of parameters involved in environment

Parameter	Range	Meaning
D_{LOS}	$\left[D_{\text{LOS}}^{\min}, D_{\text{LOS}}^{\max}\right]$	Distance between UAV and target point
$\delta_{\psi_{\text{LOS}}}$	$[0, 2\pi)$	Relative azimuth between LOS and the nose direction of UAV
v_{UAV}	$\left[v_{\text{UAV}}^{\min}, v_{\text{UAV}}^{\max}\right]$	UAV speed
H_{UAV}	$\left[H_{\text{UAV}}^{\min}, H_{\text{UAV}}^{\max}\right]$	UAV height
A_{Bomb}	$\left[A_{\text{Bomb}}^{\min}, A_{\text{Bomb}}^{\max}\right]$	Horizontal range of bomb
N_z	$\left[-N_z^{\max}, N_z^{\max}\right]$	Steering overload of UAV

3.2 Simulation results & analysis of guidance towards area task

3.2.1 Parameters setting of algorithms

According to the training procedure of algorithm, before we start to train, some parameters should be assigned. Parameters assignments of the algorithm are shown in Table 2. Moreover, we design the structure of networks $Q(s,a|\theta_Q)$ and $\mu(s|\theta_\mu)$ which are shown in Table 3 and Table 4 respectively. In this paper, the networks are all designed to be dense network.

Table 2 Parameters assignment of algorithm for the guidance towards area task

Parameter	Value	Meaning
K	100	Policy's learning period
N	100 000	Historical buffer capacity
τ	0.01	Soft updating parameter
k	128	Size of minibatch
η_μ	0.001	Actor networks' learning rate
η_Q	0.001	Critic networks' learning rate
α	0.5	Availability exponent of PER
β_0	0.4	Initial IS exponent
M	1 000	Maximum training episodes
T	5 000	Maximum steps per episode

Table 3 Structure of critic network $Q(s,a|\theta_Q)$ for the guidance towards area task

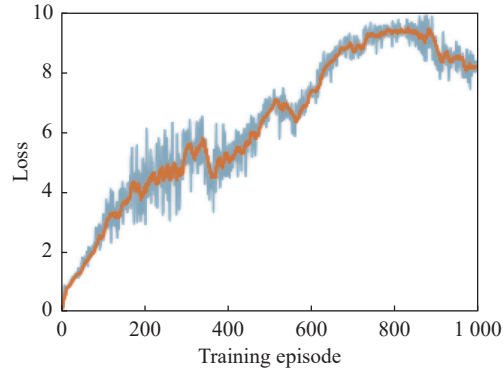
Layer name	Layer structure	
	Unit	Activation function
Input layer of state	16	ReLU
Input layer of action	16	ReLU
Hidden layer 1	32	ReLU
Hidden layer 2	64	ReLU
Hidden layer 3	32	ReLU
Output layer	1	None

Table 4 Structure of actor network $\mu(s|\theta_\mu)$ for the guidance towards area task

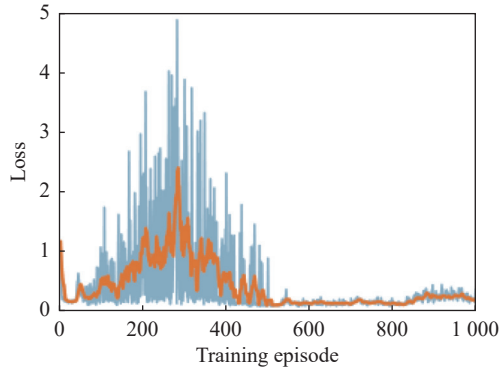
Layer name	Layer structure	
	Unit	Activation function
Input layer	16	Tanh
Hidden layer 1	32	Tanh
Hidden layer 2	64	Tanh
Hidden layer 3	32	Tanh
Output layer	1	Tanh

3.2.2 Analysis of simulation results

Using the parameters assigned above, we finish the training of networks successfully. Then, the loss diagrams of critic networks and actor networks involved in UER-DDPG without advice, PER-DDPG without advice, UER-DDPG with advice, PER-DDPG with advice for the guidance towards area task are shown in Fig. 11, Fig. 12, Fig. 13, Fig. 14, respectively.



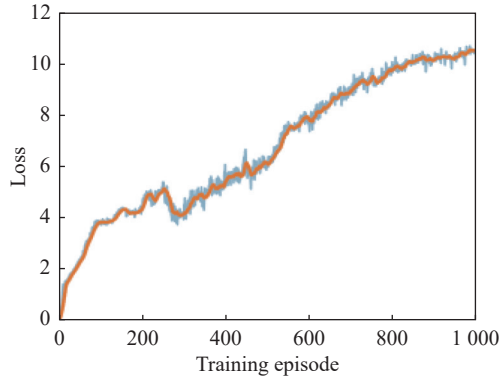
(a) Loss curve of actor network over training steps



(b) Loss curve of critic network over training steps

: Raw data; : Smoothed data.

Fig. 11 Loss curves of actor network and critic network over training steps under the setting of UER-DDPG and RS without advice in the guidance towards area task



(a) Loss curve of actor network over training steps

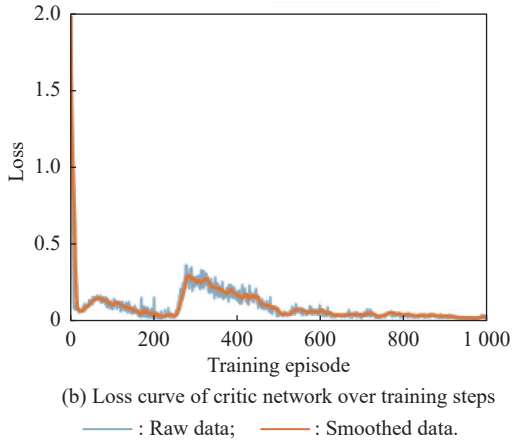


Fig. 12 Loss curves of actor network and critic network over training steps under the setting of PER-DDPG and RS without advice in the guidance towards area task

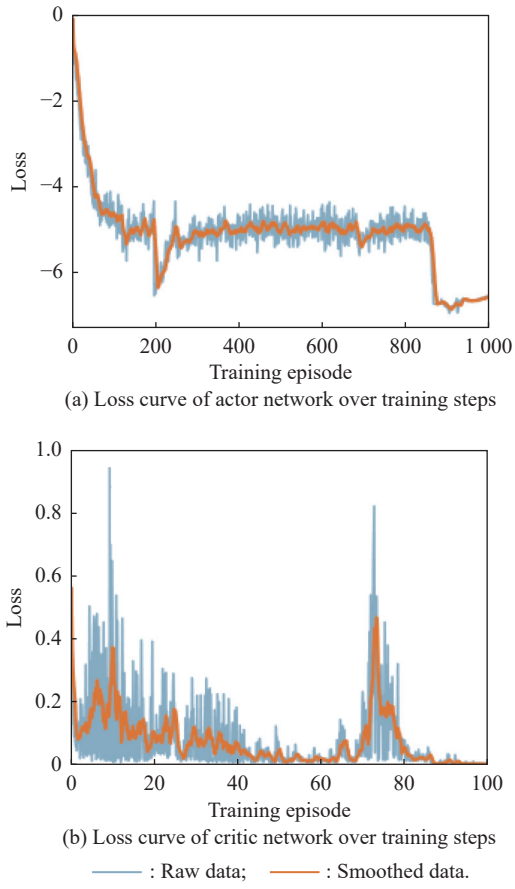


Fig. 13 Loss curves of actor network and critic network over training steps under the setting of UER-DDPG and RS with advice in the guidance towards area task

Fig. 11 and Fig. 12 show the loss curves of actor network and critic network involved in UER-DDPG and PER-DDPG without advice, respectively. The curve of actor network's loss climbs gradually over time and is stable after enough training. Meanwhile, the loss of critic network decreases gradually over time, and finally becomes stable to a small amount. In Fig. 13 and Fig. 14 similar to those algorithms without advice, the loss curves of actor network and critic network are also stable in the end by using the algorithms with advice. However, we can see that the loss of PER-DDPG converges faster than UER-DDPG and it is more stable than UER-DDPG after convergence, from not only the actor loss but the critic loss.

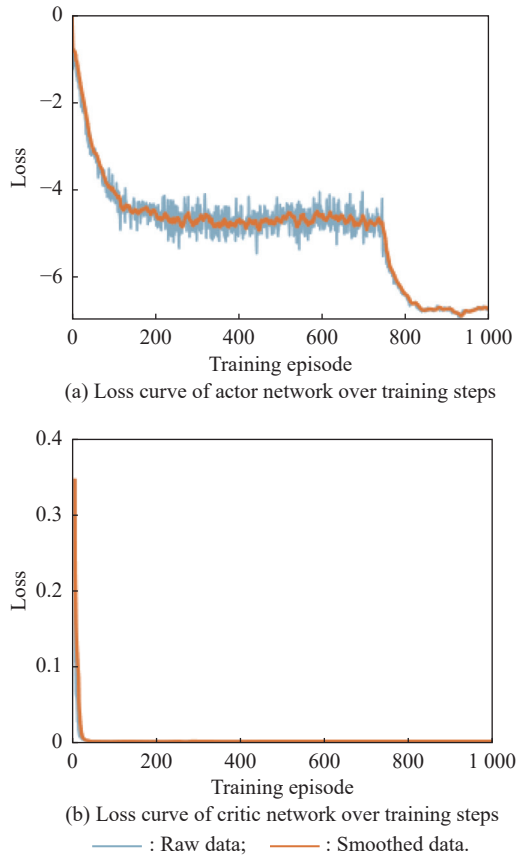
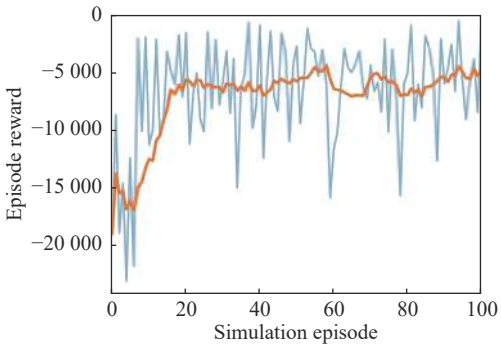
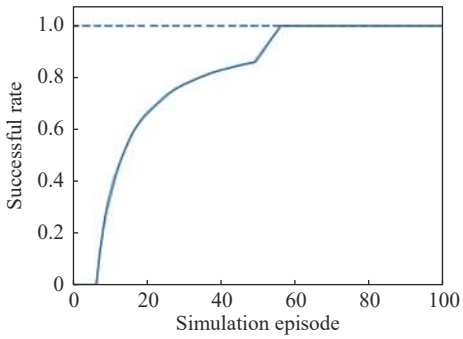


Fig. 14 Loss curves of actor network and critic network over training steps under the setting of PER-DDPG and RS with advice in the guidance towards area task

Moreover, Fig. 15, Fig. 16, Fig. 17 and Fig. 18 show that the training parameters are generated from training process over simulation episodes, including the episode rewards and successful rate under the settings of UER-DDPG without advice, PER-DDPG without advice, UER-DDPG with advice, and PER-DDPG with advice. The episode rewards refer to the accumulation of reward agent received in each episode. The successful rate refers to the ratio of successful results in the last 50 experiments.



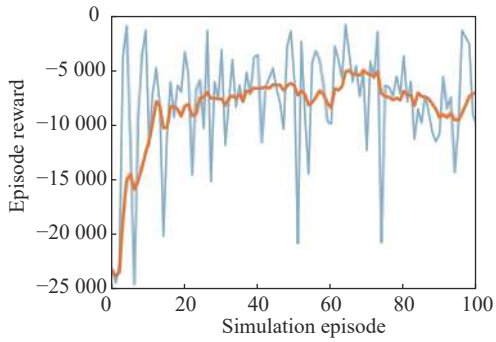
(a) Curve of cumulative rewards over simulation episodes



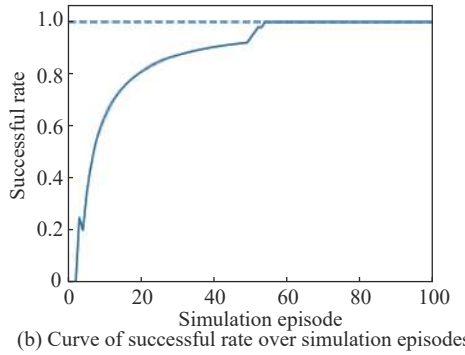
(b) Curve of successful rate over simulation episodes

: Raw data; : Smoothed data.

Fig. 15 Curves of evaluation parameters for training under the setting of UER-DDPG and RS without advice in the guidance towards areatask



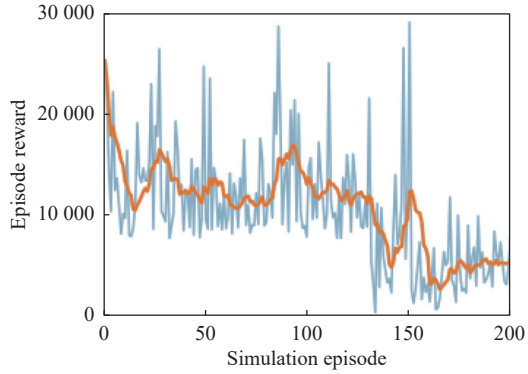
(a) Curve of cumulative rewards over simulation episodes



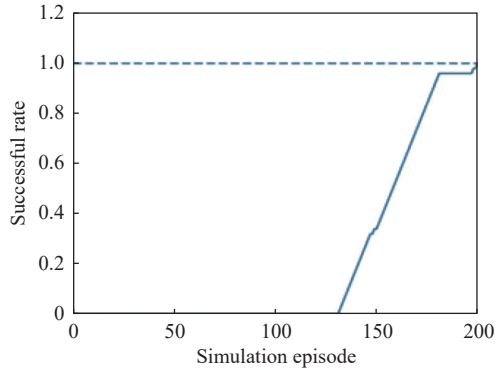
(b) Curve of successful rate over simulation episodes

: Raw data; : Smoothed data.

Fig. 16 Curves of evaluation parameters for training under the setting of PER-DDPG and RS without advice in the guidance towards areatask



(a) Curve of cumulative rewards over simulation episodes

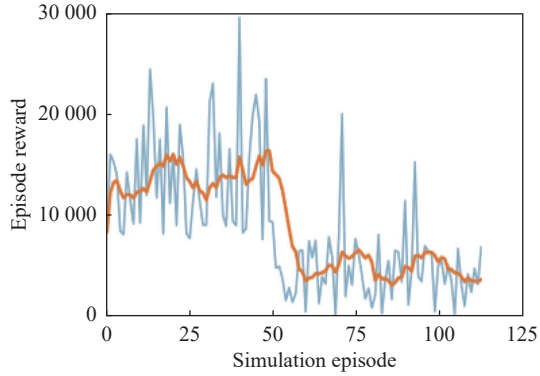


(b) Curve of successful rate over simulation episodes

: Raw data; : Smoothed data.

Fig. 17 Curves of evaluation parameters for training under the setting of UER-DDPG and RS with advice in the guidance towards area task

Fig. 15(a), Fig. 16(a), Fig. 17(a) and Fig. 18(a) are the curves of cumulative rewards over simulation episodes. They show that cumulative rewards will be stable at a deterministic value and the trend of all curves is similar. Fig. 15(b), Fig. 16(b), Fig. 17(b) and Fig. 18(b) are the curves of the successful rateover simulation episodes, which demonstrate that the parameters increase until 1.0. We can see that all the training experiments have optimized the optimal policy, but much more time is consumed during the process of training when introducing expert advice. This indicates that extra training time is needed for the agent to learn the more valuable information.



(a) Curve of cumulative rewards over simulation episodes

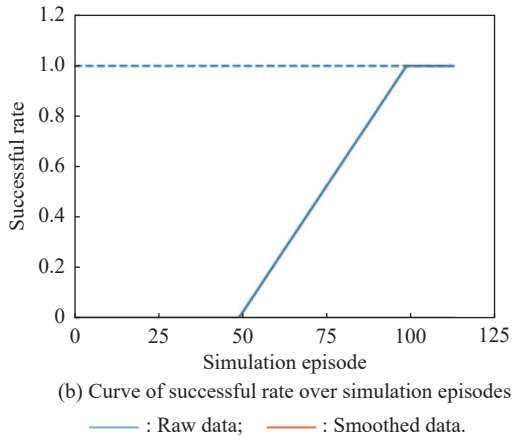


Fig. 18 Curves of evaluation parameters for training under the setting of PER-DDPG and RS with advice in the guidance towards area task

After training, we perform a group of MC experiments to evaluate the quality of trained results of the designed algorithms described above. Statistical analysis results of MC test experiments are shown in Table 5. We can see that all the four groups of experiments' results demonstrate the trained results converged to the near optimal point.

Table 5 Statistical results of MC test experiments in the guidance towards area task

Method	Number of experiments	Number of successful experiments	Successful rate
UER-DDPG and RS without advice	1 000	998	0.998
PER-DDPG and RS without advice	1 000	999	0.999
UER-DDPG and RS with advice	1 000	999	0.999
PER-DDPG and RS	1 000	999	0.999

Meanwhile, we visualize some assessment results from MC experiments, including the flight trajectory of UAV in an experiment, the action given by agent over time, and the reward agent received over time, as shown in Fig. 19, Fig. 20, Fig. 21 and Fig. 22. In Fig. 19(a), Fig. 20(a), Fig. 21(a), and Fig. 22(a), the red solid line represents the flight trajectory of UAV, the red point and the green "x" indicate the start position and the end position of UAV respectively, the blue "+" and the blue dashed circle surrounding it indicate the target position and its effective area respectively. In (b) subplots of these figures, these are the curve of action output by algorithm and the curve of rewardsover time.

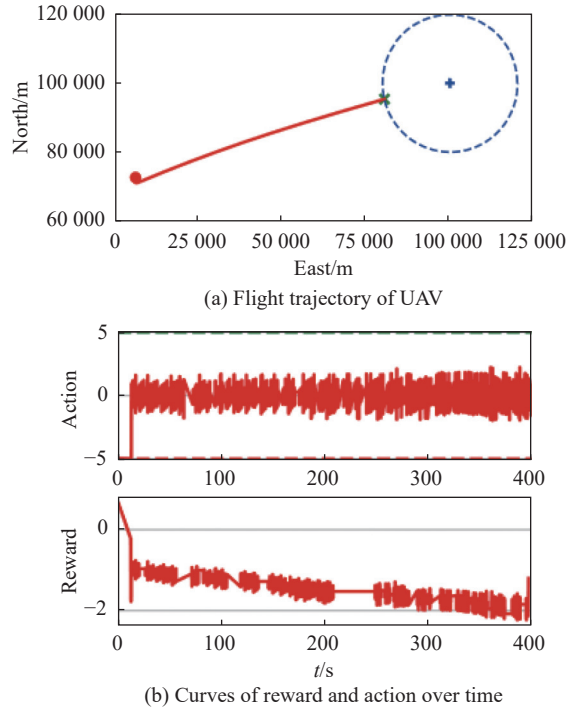


Fig. 19 Visualization of test experiment for the trained policy of UER-DDPG and RS without advice in the guidance towards area task

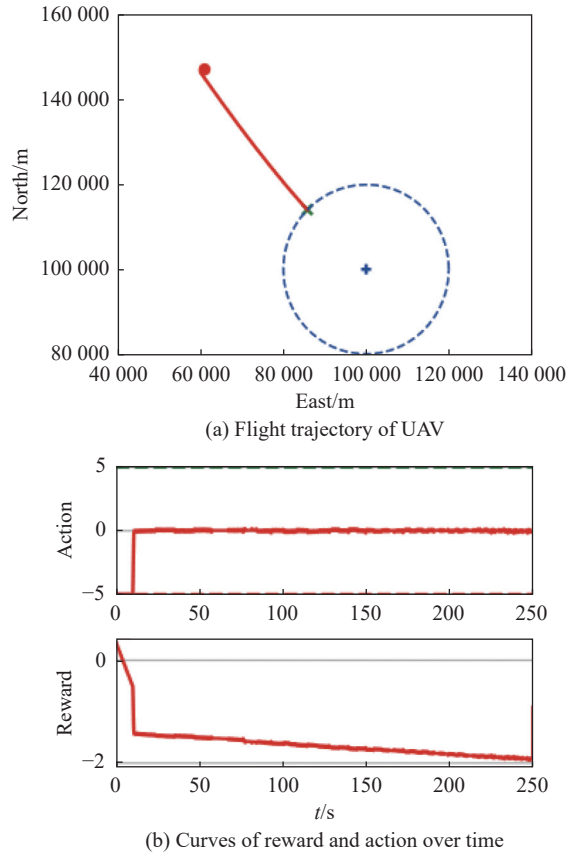


Fig. 20 Visualization of test experiment for the trained policy of PER-DDPG and RS without advice in the guidance towards area task

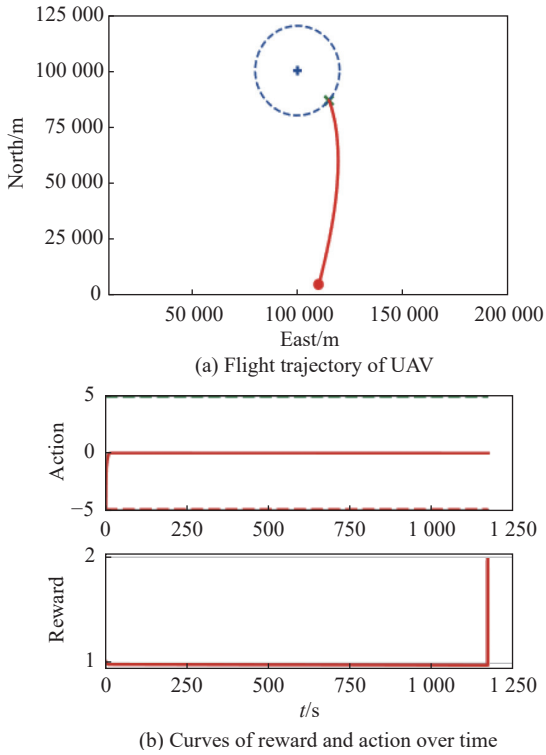


Fig. 21 Visualization of test experiment for the trained policy of UER-DDPG and RS with advice in the guidance towards area task

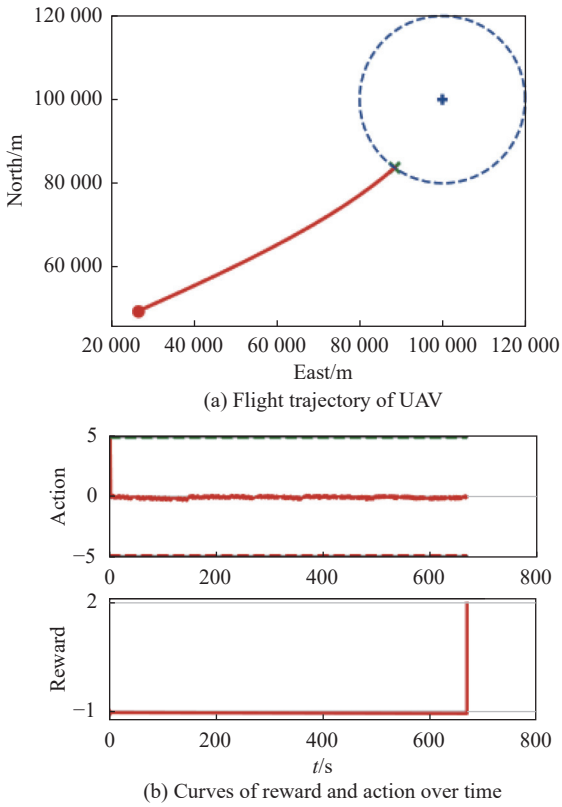


Fig. 22 Visualization of test experiment for the trained policy of PER-DDPG and RS with Advice in the guidance towards area task

In Fig. 19(a), Fig. 20(a), Fig. 21(a), and Fig. 22(a), we can see that the trained policy of each algorithm with different kind of reward function in the guidance towards area task converges to the near optimal point because of its approximate optimal performance. The UAV could enter the mission area from arbitrary position and random azimuth, and all the algorithms show good performance on this task. Meanwhile, we can see that the policy outputs a reasonable control value at different situations according to the curves of action for each algorithm. For example, when the UAV flies towards the target area, the action of policy is approximately equal to 0. If the target area is located on the front left side of UAV, the action of policy will be a negative value. On the contrary, the policy will give a positive value while the target area is located on the front right side of UAV. And the reward curve also shows the same trend which means when the UAV selects the action which makes it closer to the target area, the reward given by model will be bigger. Especially for the model improved by RS with advice, when the UAV selects the action that potentially makes bigger difference between current state and last state, it could receive a positive value, as shown in Fig. 21(b) and Fig. 22(b).

Furthermore, the trained policies show good performance in the guidance towards area task, but different kinds of shaping function produce disparate performance in terms of policy output. Obviously, the output of those algorithms with advice in Fig. 21(b) and Fig. 22(b) are much stabler than those without advice in Fig. 19(b) and Fig. 20(b), because there are not so many high-frequency fluctuations in the curves of reward and action of Fig. 21 and Fig. 22.

Thereby, it is proven that the algorithms and the modified model we design is reasonable and effective to solve the guidance towards area task and improve the autonomy of UAV during the process of adjusting attitude when it prepares to drop bomb. Furthermore, the trained results of UER-DDPG and PER-DDPG with expert advice have more superior quality in terms of policy's output than algorithms without expert advice because the output of policy involved in UER-DDPG and PER-DDPG with expert advice is smoother.

3.3 Simulation results and analysis of guidance towards specific point task

3.3.1 Parameters setting of algorithms

Similarly, before we start to train, some parameters

should be assigned. Parameters assignments of algorithm are shown in Table 6. Moreover, we design the structure of networks $Q(s,a|\theta_Q)$ and $\mu(s|\theta_\mu)$ as shown in Table 7 and Table 8 respectively. In this paper, the networks are all designed to dense network.

Table 6 Parameters assignment of algorithm for the guidance towards specific point task

Parameter	Value	Meaning
K	100	Policy's learning period
N	50 000	Historical buffer capacity
τ	0.01	Soft updating parameter
k	256	Size of minibatch
η_μ	0.001	Actor networks' learning rate
η_Q	0.001	Critic networks' learning rate
α	0.5	Availability exponent of PER
β_0	0.1	Initial IS exponent
M	3 000	Maximum training episodes
T	5 000	Maximum steps per episode

Table 7 Structure of critic network $Q(s,a|\theta_Q)$ for the guidance towards specific point task

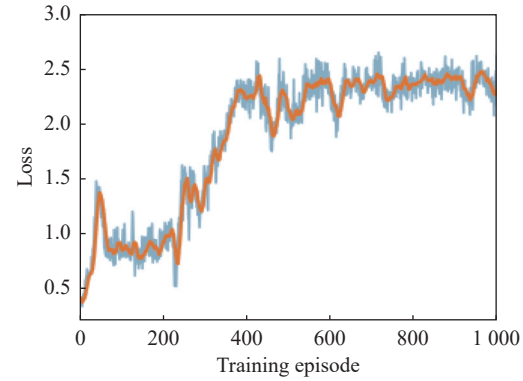
Layer name	Layer structure	
	Unit	Activation function
Input layer of state	16	ReLU
Input layer of action	16	ReLU
Hidden layer 1	32	ReLU
Hidden layer 2	64	ReLU
Hidden layer 3	32	ReLU
Output layer	1	None

Table 8 Structure of actor network $\mu(s|\theta_\mu)$ for the guidance towards specific point task

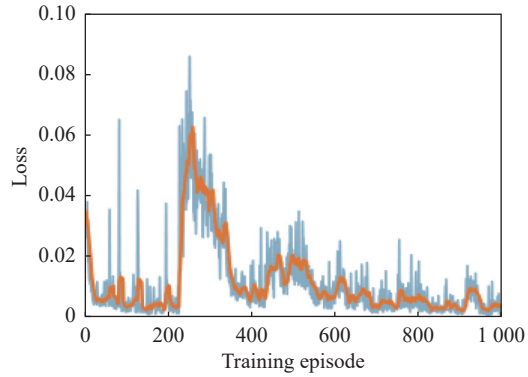
Layer name	Layer structure	
	Unit	Activation function
Input layer	16	Tanh
Hidden layer 1	32	Tanh
Hidden layer 2	64	Tanh
Hidden layer 3	32	Tanh
Output layer	1	Tanh

3.3.2 Analysis of simulation results

Similar to the guidance towards area task, we also obtain reasonable results after training. As shown in Fig. 23, Fig. 24, Fig. 25 and Fig. 26, these are the loss diagrams of critic networks and actor networks involved in UER-DDPG without advice, PER-DDPG without advice, UER-DDPG with advice, and PER-DDPG with advice for the guidance towards specific point task, respectively.



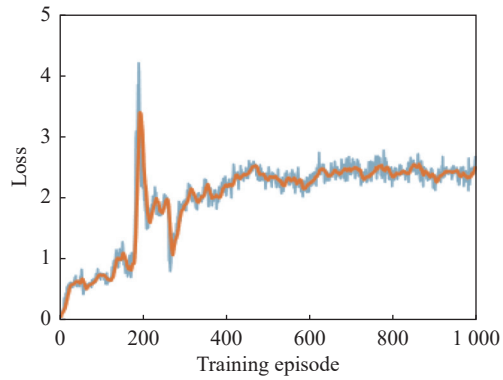
(a) Loss curve of actor network over training steps



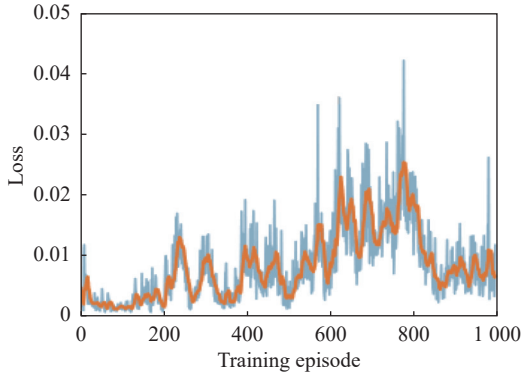
(b) Loss curve of critic network over training steps

: Raw data; : Smoothed data.

Fig. 23 Loss curves of actor network and critic network over training steps under the setting of UER-DDPG and RS without advice in the guidance towards specific point task



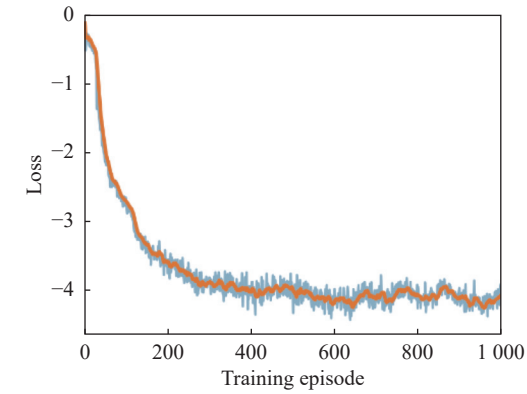
(a) Loss curve of actor network over training steps



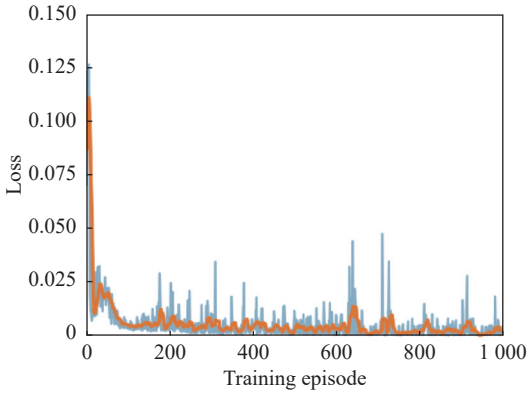
(b) Loss curve of critic network over training steps

: Raw data; : Smoothed data.

Fig. 24 Loss curves of actor network and critic network over training steps under the setting of PER-DDPG and RS without advice in theguidance towards specific point task



(a) Loss curve of actor network over training steps

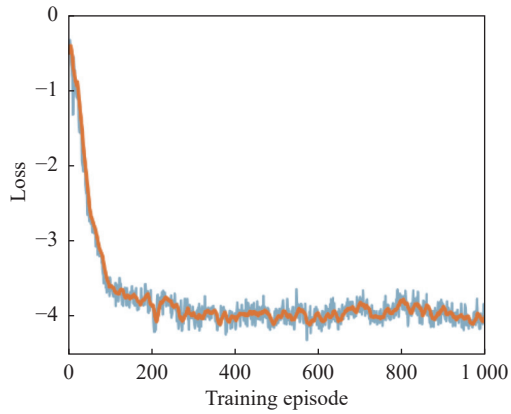


(b) Loss curve of critic network over training steps

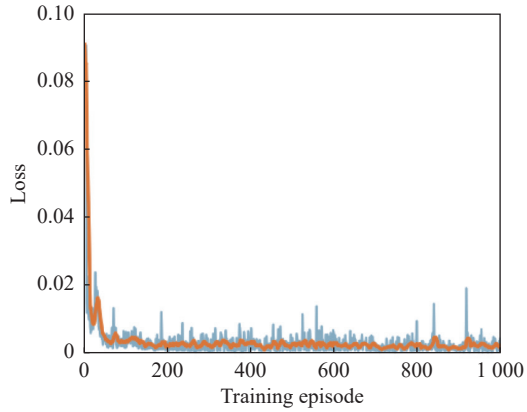
: Raw data; : Smoothed data.

Fig. 25 Loss curves of actor network and critic network over training steps under the setting of UER-DDPG and RS with advice in the guidance towards specific point task

In Fig. 23(a), Fig. 24(a), Fig. 25(a), and Fig. 26(a), the loss curves of actor network are presented, which show that actor's losses increase or decrease until stabilizing at a deterministic value. In Fig. 23(b), Fig. 24(b), Fig. 25(b), and Fig. 26(b), the loss curves of critic network are shown, which demonstrate that the critic networks stabilize at a good value after lots of training, though there are some peaks during training. Among these figures, we could find that all the algorithms under different reward functions converge to near optimal point and the loss of PER-DDPG is more stable than UER-DDPG after convergence in terms of the smoothness of loss curve.



(a) Loss curve of actor network over training steps

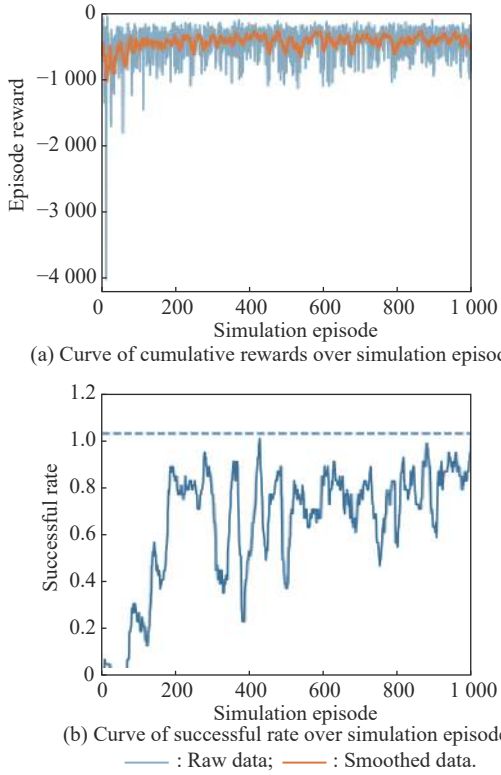


(b) Loss curve of critic network over training steps

: Raw data; : Smoothed data.

Fig. 26 Loss curves of actor network and critic network over training steps under the setting of PER-DDPG and RS with advice in the guidance towards specific point task

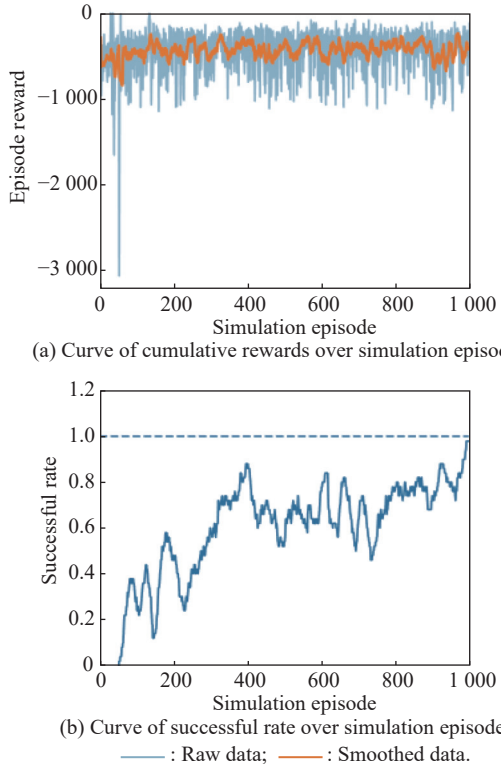
Moreover, Fig. 27, Fig. 28, Fig. 29, Fig. 30 show the training parameters generated from training process over simulation episodes, including the episode rewards and successful rate under the settings of UER-DDPG without advice, PER-DDPG without advice, UER-DDPG with advice, and PER-DDPG with advice.



(a) Curve of cumulative rewards over simulation episodes

(b) Curve of successful rate over simulation episodes

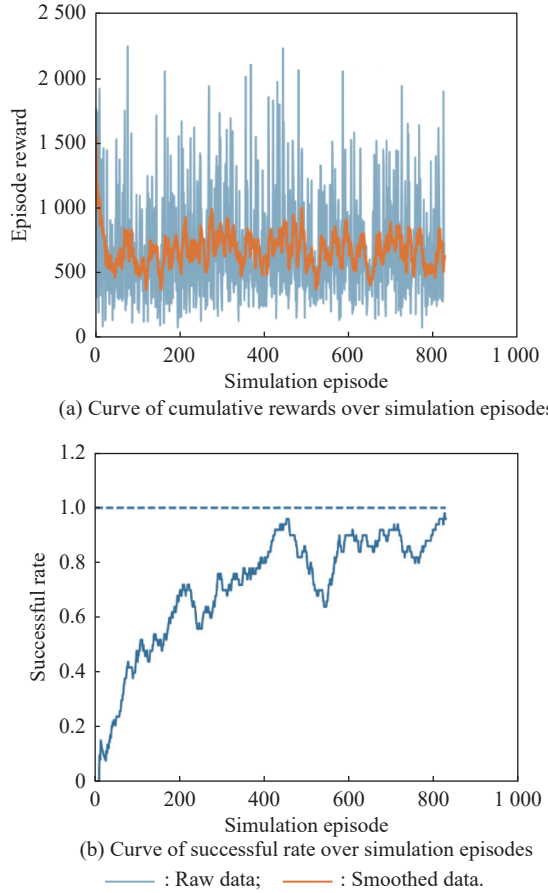
Fig. 27 Curves of evaluation parameters for training under the setting of UER-DDPG and RS without advice in the guidance towards specific point task



(a) Curve of cumulative rewards over simulation episodes

(b) Curve of successful rate over simulation episodes

Fig. 28 Curves of evaluation parameters for training under the setting of PER-DDPG and RS without advice in the guidance towards specific point task

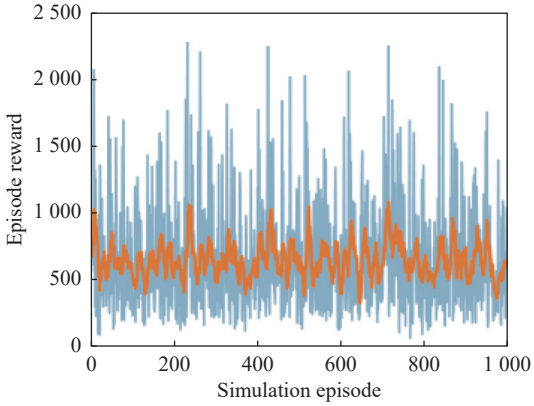


(a) Curve of cumulative rewards over simulation episodes

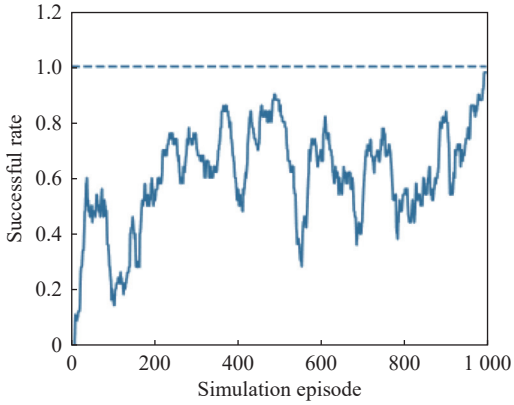
(b) Curve of successful rate over simulation episodes

Fig. 29 Curves of evaluation parameters for training under the setting of UER-DDPG and RS with advice in the guidance towards specificpoint task

Fig. 27(a), Fig. 28(a), Fig. 29(a), and Fig. 30(a) show that the episode rewards increase gradually till a fixed value. Fig. 27(b), Fig. 28(b), Fig. 29(b), and Fig. 30(b) are the curves of successful rate, which show that though the training process is fluctuant, the actor and critic of each algorithm converge at an excellent level. Among these results, we can obtain the optimal policy in all the training experiments, but much more time is consumed during the process of training when introducing expert advice due to similar reasons. In addition, algorithms are required to operate UAV more accurately in guidance towards specific point task because the impact point will be target point far away when the agent only slightly adjusts the azimuth of UAV due to the range of bomb. Thereby, it is difficult for algorithms to converge to the optimal point due to overestimate bias and variance.



(a) Curve of cumulative rewards over simulation episodes



(b) Curve of successful rate over simulation episodes

: Raw data; : Smoothed data.

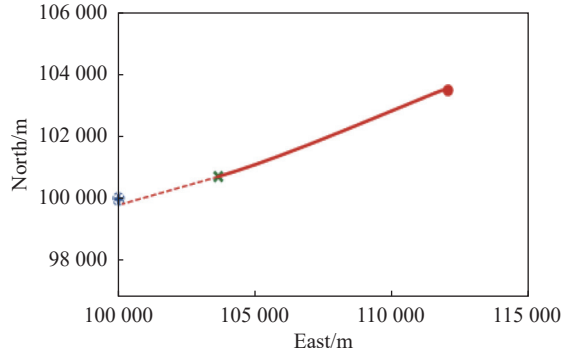
Fig. 30 Curves of evaluation parameters for training under the setting of PER-DDPG and RS with advice in the guidance towards specific point task

In the same way, we also design a group of MC experiments to assess the quality of trained results of algorithms above in the guidance towards specific point task. Statistical analysis results of MC test experiments are shown in Table 9. We can see that all four groups of experimental results demonstrate that the trained results converge to the near optimal point and have achieved an available and satisfied level.

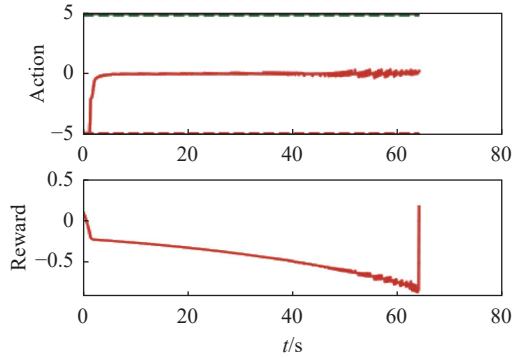
Table 9 Statistical results of MC test experiments in the guidance towards specific point task

Method	Number of experiments	Number of successful experiments	Successful rate
UER-DDPG and RS without advice	1 000	743	0.743
PER-DDPG and RS without advice	1 000	893	0.893
UER-DDPG and RS with advice	1 000	957	0.957
PER-DDPG and RS	1 000	943	0.943

Simultaneously, we visualize some assessment results from MC experiments, including the flight trajectory of UAV in an experiment, the action given by agent over time, and the reward agent received over time, as shown in Figs. 31−34. In (a) subplots of each figure, the red solid line represents the flight trajectory of UAV, the red dashed line represents the trajectory of bomb on the horizontal plane, the red point and the green "x" indicate the start position and the end position of UAV respectively, and the blue "+" and the blue dashed circle surrounding it indicate the target position and its effective area separately.

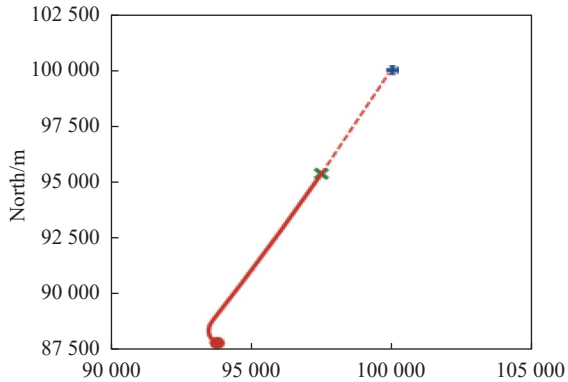


(a) Flight trajectory of UAV

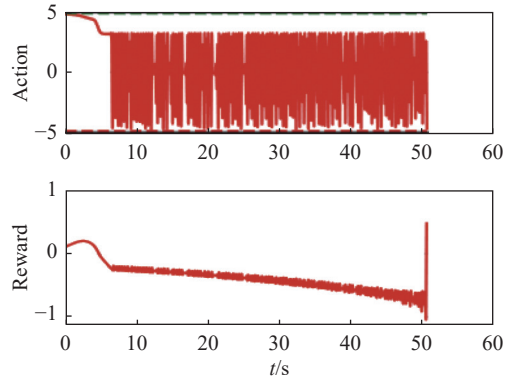


(b) Curves of reward and action over time

Fig. 31 Visualization of test experiment for the trained policy of UER-DDPG and RS without advice in the guidance towards specific point task

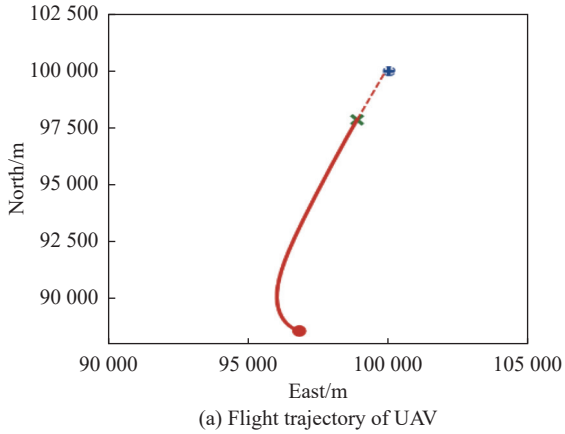


(a) Flight trajectory of UAV

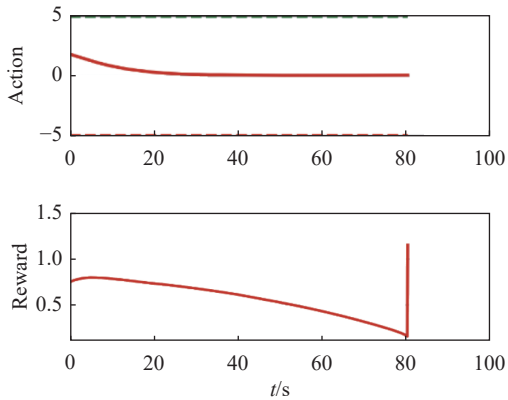


(b) Curves of reward and action over time

Fig. 32 Visualization of test experiment for the trained policy of PER-DDPG and RS without advice in the guidance towards specific point task



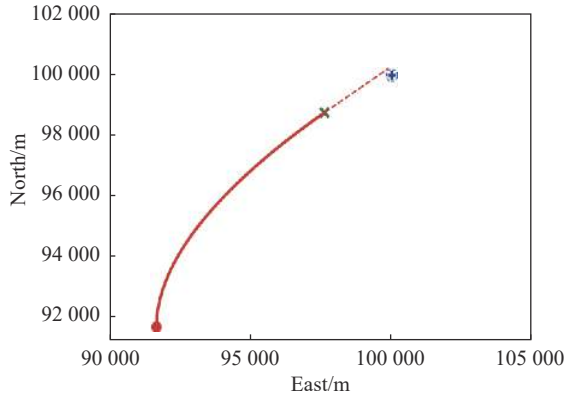
(a) Flight trajectory of UAV



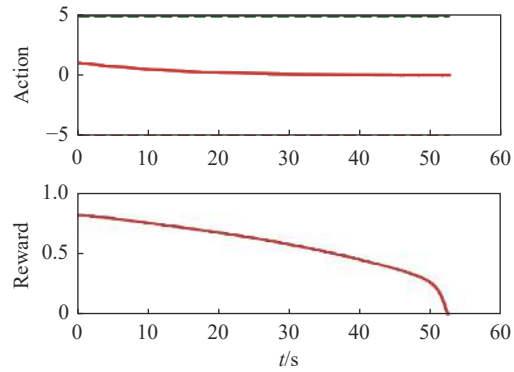
(b) Curves of reward and action over time

Fig. 33 Visualization of test experiment for the trained policy of UER-DDPG and RS with advice in the guidance towards specific point task

Similar to guidance towards area task, the proposed algorithms show a good performance in the guidance towards specific point task. As shown in Figs. 31−34, the policy converges to the near optimal position after training. In Fig. 31(a), Fig. 32(a), Fig. 33(a), and Fig. 34(a), the UAV reaches an appropriate position to release bomb by giving a reasonable control value to adjust the attitude of UAV. For instance, if the desired impact point is located at the front left side of UAV, the policy will give a negative action. On the contrary, the action output by policy will become positive when the point is located at the right left side of UAV. We could find these results from the curve of action in Fig. 31(b), Fig. 32(b), Fig. 33(b), and Fig. 34(b).



(a) Flight trajectory of UAV



(b) Curves of reward and action over time

Fig. 34 Visualization of test experiment for the trained policy of PER-DDPG and RS with advice in the guidance towards specific point task

Moreover, we could find the good performance of trained policies in the guidance towards specific point task, but different kinds of reward shaping methods cause the difference of policy in terms of output stability and robustness. Obviously, the output of policies trained based on RS with advice is much stabler than those based on RS without advice, because there is not much irregular noise in the output value of policy shown in Fig. 33(b) and Fig. 34(b), compared with Fig. 31(b) and Fig. 32(b).

Thereby, it is proved that the algorithms and the modified model we design is reasonable and effective to solve the guidance towards specific point task and enhance the autonomy of UAV during the process of aiming at the

target when it prepares to release bomb. Furthermore, the performance of trained results of UER-DDPG and PER-DDPG with expert advice is similar to the results in the guidance towards area task. The output of algorithms introducing expert advice is smoother than that of algorithms without expert advice. This superiority could help the policy trained by the DRL-based algorithm be transferred to the real world, because the high frequency vibration is unbearable for machine.

4. Conclusions

In the present work, we refine and describe the guidance towards area task and the guidance towards specific task in air-delivery. According to the definitions of problems, we propose the UAV maneuvering decision-making algorithm based on DRL to execute air-delivery mission autonomously. Among this work, we design and construct the UAV maneuvering decision-making model based on MDPs, consisting of state space, action space and reward function of each task. Then, we present the UAV maneuvering decision-making algorithm based on PER-DDPG, in which the PER sampling method improves the availability of historical data during the training process. Specifically, we propose a construction method of modified reward function that took domain knowledge and expert advice into account to improve the inference quality of trained policy network.

Meanwhile, we design extensive experiments to verify the performance of proposed algorithms and model. The parameters generated from training process show that PER method could help algorithm converge more quickly than UER and the trained results are stabler than that using UER. Furthermore, the MC experiments results demonstrate that the modified reward function involving expert advice can significantly improve the quality of policy trained by algorithms and has better performance to achieve accurate operation.

In the future, we will consider the influence of information dimension loss, which means environment is partially observed. And we will extend the proposed algorithm to manipulate real UAVs in a 3D environment while performing special missions.

References

[1] HALUDER H, BREZAK M, PETROVIC I, et al. UAV-enabled intelligent transportation systems for the smart city: applications and challenges. IEEE Communications Magazine, 2017, 55(3): 22–28.

[2] MATHISEN S G, LEIRA F S, HELGESEN H H, et al. Autonomous ballistic airdrop of objects from a small fixed-wing unmanned aerial vehicle. Autonomous Robots, 2020, 44: 859–875.

[3] LYU X, LI X B, DANG D L, et al. Unmanned aerial vehicle (UAV) remote sensing in grassland ecosystem monitoring: a systematic review. Remote Sensing, 2022, 14(5): 1096.

[4] UKAEGBU U, TARTIBU L, OKWU M. Unmanned aerial vehicles for the future: classification, challenges, and opportunities. Proc. of the International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems, 2021. DOI: 10.1109/icABXD51485.2021.9519367.

[5] SEUNGHYEON L, YOUNGKEUN S, SUNG-HO K. Feasibility analyses of real-time detection of wildlife using UAV-derived thermal and rgb images. Remote Sensing, 2021, 13(11): 2169.

[6] LI K, ZHANG K, ZHANG Z C, et al. A UAV maneuver decision-making algorithm for autonomous airdrop based on deep reinforcement learning. Sensors, 2021, 21(6): 2233.

[7] ZHANG K, LI K, HE J L, et al. A UAV autonomous maneuver decision-making algorithm for route guidance. Proc. of the International Conference on Unmanned Aircraft Systems, 2020: 17−25.

[8] WAN K F, LI B, GAO X G, et al. A learning-based flexible autonomous motion control method for UAV in dynamic unknown environments. Journal of Systems Engineering and Electronics, 2021, 32(6): 1490–1508.

[9] ZHANG J D, YANG Q M, SHI G Q, et al. UAV cooperative air combat maneuver decision based on multi-agent reinforcement learning. Journal of Systems Engineering and Electronics, 2021, 32(6): 1421–1438.

[10] YANG L, QI J T, XIAO J Z, et al. A literature review of UAV 3D path planning. Proc. of the 11th World Congress on Intelligent Control and Automation, 2014: 2376−2381.

[11] HRVOJE K, MISEL B, IVAN P. A visibility graph based method for path planning in dynamic environments. Proc. of the 34th International Convention MIPRO, 2011: 717−721.

[12] SUN Q P, LI M, WANG T H, et al. UAV path planning based on improved rapidly-exploring random tree. Proc. of the Chinese Control and Decision Conference, 2018: 6420−6424.

[13] YAN F, LIN Y S, XIAO J Z. Path planning in complex 3D environments using a probabilistic roadmap method. International Journal of Automation and Computing, 2013, 10(6): 525–533.

[14] FAN H T, TSUNG T L, CHO H L, et al. A star search algorithm for civil UAV path planning with 3G communication. Proc. of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing, 2014: 942−945.

[15] MENG B B. UAV path planning based on bidirectional sparse A* search algorithm. Proc. of the International Conference on Intelligent Computation Technology and Automation, 2010: 1106−1109.

[16] DAVE F, ANTHONY S. Using interpolation to improve path planning: the Field D* algorithm. Journal of Field Robotics, 2006, 23(2): 79–101.

[17] JESIMAR D S, MARCIO D S, CLAUDIO F M T, et al. Heuristic and genetic algorithm approaches for UAV path planning under critical situation. International Journal on Artificial Intelligence Tools, 2017, 26(1): 1760008.

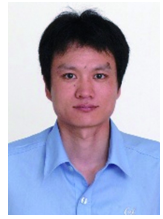
[18] LEE H, KIM H J. Trajectory tracking control of multirotors from modelling to experiments: a survey. International Journal of Control, Automation and Systems, 2017, 15: 281–292.

[19] SEBASTIAN B, BEN-TZVI P. Physics based path planning for autonomous tracked vehicle in challenging terrain. Journal of Intelligent & Robotic Systems, 2019, 95: 511–526.

[20] XU Z, ZHANG E Z, CHEN Q W. Rotary unmanned aerial

vehicles path planning in rough terrain based on multi-objective particle swarm optimization. Journal of Systems Engineering and Electronics, 2020, 31(1): 130–141.
[21] OBERMEYER K J. Path planning for a UAV performing reconnaissance of static ground targets in terrain. Proc. of the AIAA Guidance, Navigation, and Control Conference, 2009: 5888.
[22] FRANCOIS-LAVET V, HENDERSON P, ISLAM R, et al. An introduction to deep reinforcement learning. Foundations and Trends in Machine Learning, 2018, 11(3/4): 219–354.
[23] VOLODYMYR M, KORAY K, DAVID S, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518(7540): 529–533.
[24] TIMOTHY P L, JONATHAN J H, ALEXANDER P, et al. Continuous control with deep reinforcement learning. https://arxiv.org/abs/1509.02971.
[25] TOM S, JOHN Q, IOANNIS A, et al. Prioritized experience replay. https://arxiv.org/abs/1511.05952.
[26] PIAO H Y, SUN Z X, MENG G L, et al. Beyond-visual-range air combat tactics auto-generation by reinforcement learning. Proc. of the International Joint Conference on Neural Networks, 2020. DOI: 10.1109/IJCNN48605.2020.9207088.
[27] SUN Z K, PIAO H Y, YANG Z, et al. Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play. Engineering Applications of Artificial Intelligence, 2021, 98: 104112.
[28] SHANKARACHARY R, EDWIN K P C. UAV path planning in a dynamic environment via partially observable Markov decision process. IEEE Trans. on Aerospace and Electronic Systems, 2013, 49(4): 2397–2412.
[29] MARTIJN V O, MARCO W. Reinforcement learning. Cham: Springer, 2012: 3−42.
[30] MASSENGILL H J. A technique for predicting aircraft flowfield effects upon an unguided bomb ballistic trajectory and comparison with flight test results. Proc. of the 31st Aerospace Sciences Meeting, 1993: 856.
[31] MASSENGILL H J. A comparison of simulated and flight test ballistic trajectories for stores released from an aircraft in flight. Proc. of the 35th Aerospace Sciences Meeting and Exhibit, 1997: 926.
[32] BABAK B, NASSER M, MOZAYANI N et al. A new potential-based reward shaping for reinforcement learning agent. Proc. of the IEEE 13th Annual Computing and Communication Workshop and Conference, 2023. DOI: 10.1109/CCWC57344.2023.10099211.
[33] ANDREW Y N, DAISHI H, STUART R. Policy invariance under reward transformations: theory and application to reward shaping. Proc. of the International Conference on Machine Learning, 1999: 278−287.
[34] ERIC W, GARRISON W C, CHARLES E. Principled methods for advising reinforcement learning agents. Proc. of the 20th International Conference on Machine Learning, 2003: 792−799.

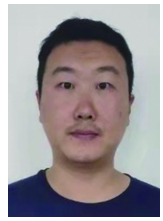
Biographies



ZHAN Guang was born in 1979. He received his B.S. degree from Beihang University, China, in 2001. He received his M.S. degree from Beihang University, China, in 2004. He is currently pursuing his Ph.D. degree in electronics and information technology with the School of Electronic Information at Northwestern Polytechnical University. His research interests are unmanned aerial vehicle and cluster intelligence.
E-mail: zhanguang@mail.nwpu.edu.cn

ZHANG Kun was born in 1982. He received his Ph.D. degree from Northwestern Polytechnical University in 2010. He is an associate professor in Northwestern Polytechnical University. His research interests are intelligent air combat, intelligent control and decision-making, integrated avionics system simulation and testing, and advanced control theory and application.
E-mail: kunzhang@nwpu.edu.cn

LI Ke was born in 1996. He received his B.S. degree from Northwestern Polytechnical University, Xi'an, China in 2018. Currently, he is pursuing his Ph.D. degree at Northwestern Polytechnical University. His research interests are autonomous flight of unmanned aerial vehicle, and unmanned aerial vehicle swarm control.
E-mail: keli_iat@mail.nwpu.edu.cn



PIAO Haiyin was born in 1984. He received his M.S. degree in computer science from Dalian University of Technology, China, in 2010. He is currently working toward his Ph.D. degree in School of Electronics and Information, Northwestern Polytechnical University, China. His current research interests include deep learning, multi-agent reinforcement learning, and game theory with particular attention to aerospace applications.
E-mail: haiyinpiao@mail.nwpu.edu.cn