Feng *et al*. *Intell Robot* 2023;3(2):161-75
DOI: 10.20517/ir.2023.10

Intelligence & Robotics

Research Article

Open Access

Check for updates

Reinforcement learning with parameterized action space and sparse reward for UAV navigation

Shiying Feng[1], Xiaofeng Li[2], Lu Ren[2], Shuiqing Xu[3]

[1]Institute of Physical Science and Information Technology, Anhui University, Hefei 230601, Anhui, China.
[2]School of Artificial Intelligence, Anhui University, Hefei 230601, Anhui, China.
[3]School of Electrical Engineering and Automation, Hefei University of Technology, Hefei 230009, Anhui, China.

Correspondence to: Lu Ren, School of Artificial Intelligence, Anhui University, Hefei 230601, Anhui, China. E-mail: penny_lu@ahu.edu.cn

How to cite this article: Feng S, Li X, Ren L, Xu S. Reinforcement learning with parameterized action space and sparse reward for UAV navigation. *Intell Robot* 2023;3:161-75. http://dx.doi.org/10.20517/ir.2023.10

Received: 23 Feb 2023 **First Decision:** 27 Apr 2023 **Revised:** 11 May 2023 **Accepted:** 18 May 2023 **Published:** 27 Jun 2023

Academic Editor: Simon X. Yang **Copy Editor:** Yanbing Bai **Production Editor:** Yanbing Bai

Abstract

Autonomous navigation of unmanned aerial vehicles (UAVs) is widely used in building rescue systems. As the complexity of the task increases, traditional methods based on environment models are hard to apply. In this paper, a reinforcement learning (RL) algorithm is proposed to solve the UAV navigation problem. The UAV navigation task is modeled as a Markov Decision Process (MDP) with parameterized actions. In addition, the sparse reward problem is also taken into account. To address these issues, we develop the HER-MPDQN by combining Multi-Pass Deep Q-Network (MP-DQN) and Hindsight Experience Replay (HER). Two UAV navigation simulation environments with progressive difficulty are constructed to evaluate our method. The results show that HER-MPDQN outperforms other baselines in relatively simple tasks. Especially for complex tasks involving relay operations, only our method can achieve satisfactory performance.

Keywords: Deep reinforcement learning, parameterized action space, sparse reward

1. INTRODUCTION

In recent years, unmanned aerial vehicles (UAVs) have been widely used in emergency rescue fields within the framework of Industry 4.0[1–3]. The key technology behind these applications is UAV attitude control[4,5] and autonomous navigation[6]. Traditional approaches to address navigation challenges using modeling techniques[7–9] and simultaneously localization-and-mapping techniques[10–12]. While these methods have demon-

© The Author(s) 2023. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, sharing, adaptation, distribution and reproduction in any medium or format, for any purpose, even commercially, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

strated satisfactory performance, they heavily rely on prior knowledge of the environment, limiting their applicability in complex and dynamic navigation scenarios.

Deep reinforcement learning (DRL) has developed rapidly and achieved remarkable success in recent years. DRL aims at deriving a policy by maximizing a long-term cumulated reward of a Markov decision process (MDP). MDP characterizes a process in which an agent at some state takes action, transits to another state, and obtains rewards. For continuous decision problems, Silver *et al*.[13] develop a hybrid DRL system that defeated a human world champion in Go. In robotics, DRL successfully solves optimal control problems[14,15]. For this reason, researchers turn their attention to reinforcement learning (RL)-based methods. For instance, Yan *et al*.[16] propose an improved Q-learning algorithm to handle the path planning problem without prior knowledge. However, it is difficult to deal with tasks in the real world because only discrete action sets are considered. In contrast, Bouhamed *et al*.[17] propose a path planning framework for handling continuous actions based on DRL.

Recent work on UAV navigation based on DRL has been successful, but there are still some potential issues that have been overlooked. On the one hand, the behavior of UAVs is usually defined as single-type actions in existing studies. However, UAVs often need both discrete decision-making and continuous control when performing tasks. In[18], Masson *et al*. call the action space consisting of two control signals as parameterized action space. Hausknecht and Stone[19] deal with this problem by relaxing the parameterized action space into a continuous one. However, they do not exploit the structural information of this action space. For that matter, Xiong *et al*.[20] propose Parametrized Deep Q-Networks (P-DQN), which can directly learn from parameterized actions. Since P-DQN couples all continuous parameters to each discrete action, the prediction of the Q network is influenced by unrelated parameters. In[21], Bester *et al*. develop the Multi-Pass Deep Q-Network (MP-DQN) algorithm by changing the Q network architecture of P-DQN to eliminate this effect.

On the other hand, the studies above train agents with custom reward functions to speed up network convergence[22,23]. However, overly complex reward signals may cause the agent to get stuck in local optima. Moreover, it is hard to give correct rewards according to whether the UAV is moving away from the target due to the presence of obstacles. This problem can be avoided by adopting more general sparse reward schemes. While defining sparse reward is simple, the potential learning problem is much harder to solve (i.e., lack of intermediate rewards hinders the learning of agents). Numerous studies propose curiosity-based approaches to encourage agents to explore previously rare states[24,25]. These methods introduce additional fitting models of environmental dynamics to measure curiosity. However, the model will reduce efficiency when the dynamics are unpredictable. Andrychowicz *et al*.[26] develop the Hindsight Experience Replay (HER) algorithm by introducing the goal mechanism. The effectiveness of HER has been demonstrated in many robotics applications with sparse reward[27,28].

Parameterized action space and sparse reward inspire us to rethink new forms of RL problems. In this paper, our contributions can be summarized as follows:

1) We model UAV navigation as a parameterized action MDP to better suit the task requirements. At the same time, sparse rewards are considered to improve the generality of the algorithm in various tasks. To handle these challenges, an off-policy algorithm called HER-MPDQN is developed by incorporating HER with MP-DQN.
2) To address the issue of extended invalid experiences encountered in traditional methods, we propose a goal-switching mechanism. This mechanism effectively reduces the invalid expansion for experience and improves the rationality of the expansion experience.
3) We compare our algorithm with baselines in experiments with high randomness and varying difficulty. Experimental results show that our method is capable of learning better policies in solving navigation tasks

with sparse rewards. It can be successfully generalized to any position in space and significantly outperforms existing RL algorithms.

The rest of this article is outlined below. PAMDP and the sparse reward problem are described in Section 2. In Section 3, the navigation problem is modeled as a PAMDP and uses a sparse reward scheme. Our proposal to address the problem is elaborated in Section 4. In Section 5, we compare HER-MPDQN with other baselines in two UAV navigation simulation environments, followed by our discussion in Section 6 and conclusions in Section 7.

2. BACKGROUND

In this section, we briefly introduce MDP and PAMDP, followed by the issue of sparse reward.

2.1. MDP and PAMDP

MDP is built based on a set of interactive objects, namely agents and environments. MDP consists of a state space S, an action space A, a state transition probability distribution $P(s_{t+1}|s_t, a_t)$, and a reward function $R : r_t = r(s_t, a_t)$. Where P satisfies the Markov property, and R represents the immediate reward r_t obtained by executing an action a_t in a given state s_t. RL, which learns the optimal policy based on trial and error, provides a way to solve the MDP problem. The policies learned based on RL are divided into deterministic policy and stochastic policy. For discrete action spaces, the deterministic policy is expressed as $a_t = \pi(s_t)$, which means that a certain action a_t can be obtained for a given state s_t. The stochastic policy is denoted as $a_t \sim \mu(\cdot|s_t)$, which means to select an action a_t from the probability distribution after a given state s_t. If the action space is continuous, the corresponding policies above will be parameterized as functions $a_t = \pi(s_t, \theta)$ and $a_t \sim \mu(\cdot|s_t, \theta)$, where θ refers to the parameters of the function. To uniform representation, we all implicitly mean that policy π (or μ) is a function of θ, and all the gradients are with respect to θ in the following contexts. RL involves estimating state-value functions $V(s)$ and action-value functions $Q(s, a)$. Taking deterministic policy as an example, the state-value function is defined as

$$V_\pi(s_t) = \mathbb{E}_\pi \left[\sum_{m=0}^{T} \gamma^m r(s_{t+m}, a_{t+m}) | s_t \right], \tag{1}$$

where $\gamma \in [0, 1]$ is a discount factor, and the action-value function is defined as

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{m=0}^{T} \gamma^m r(s_{t+m}, a_{t+m}) | s_t, a_t \right]. \tag{2}$$

The agent learns an optimal policy by maximizing the expected discounted reward (target function) as follows

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right]. \tag{3}$$

If π is a stochastic policy, the corresponding state-value function, action-value function, and target function can be obtained by replacing $a_t = \pi(s_t)$ by $a_t \sim \mu(\cdot|s_t)$ in (1), (2), and (3).

PAMDP is an extension of MDP on the action space, allowing decision-making using parameterized actions. Parameterized actions flexibly integrate discrete actions and continuous actions and provide richer expressiveness. In tasks such as UAV navigation, which demand precise parameter control, using parameterized actions enables finer-grained control. Moreover, from an interpretability standpoint, the structural characteristics of parameterized actions make the decision-making process of agents more understandable and explainable. The action space of PAMDP can be expressed as $\mathcal{H} = \{(k, x_k) | x_k \in \mathcal{X}_k\}$ $for\ all\ k \in K$, where $k = \{1, ..., K\}$. K denotes the number of discrete actions. k refers to a specific discrete action (e.g., $k = 1$ means movement and

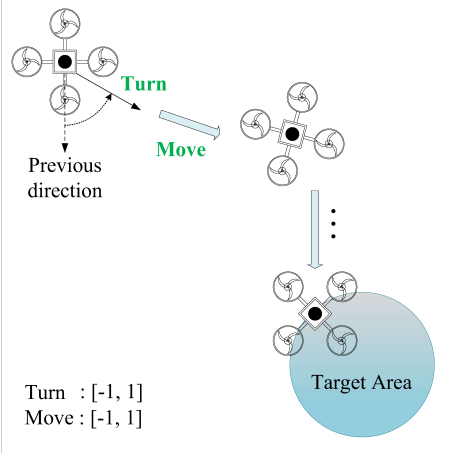

Figure 1. The direct navigation task. The UAV only needs to fly from the initial point to the target area.

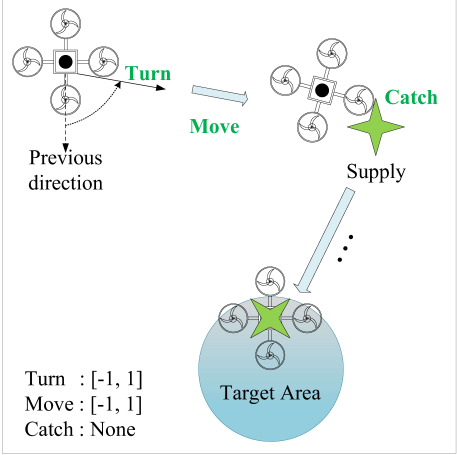

Figure 2. The relay navigation task. The UAV needs to operate on "catching supply" during the navigation process.

$k = 2$ means turning). x_k represents the continuous parameter (e.g., acceleration or angle) associated with the discrete action k. $\mathcal{X}_k$ is the set of all continuous parameters. In the PAMDP, the agent first selects a discrete action k, then obtains the corresponding x_k from $\mathcal{X}$ according to k, and finally executes the action (k, x_k). Therefore, PAMDP has subtle differences in the interaction process compared to standard MDP. Assuming that at step t, PAMDP is in state s_t, the agent executes an action by policy π: $s_t \rightarrow (k_t, x_{k_t})$ and receives an immediate reward $r(s_t, k_t, x_{k_t})$ and the next state $s_{t+1} \sim P(s_{t+1}|s_t, k_t, x_{k_t})$. The target function of the agent becomes as follows

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r(s_t, k_t, x_{k_t}) \right]. \tag{4}$$

2.2. Sparse reward problems

The sparse reward scheme is a reward mechanism that uses only binary values to indicate whether a task is eventually completed. Specifically, the agent can get a positive reward when completing the task and a negative reward during exploration. Although this mechanism reduces the effort required for human design, it brings potential learning problems. The lack of effective rewards prevents the agent from judging the pros and cons of its behavior and thus cannot optimize the policy π correctly. Under the influence of the sparse reward problem, the agent learns slowly or even fails to learn. Solving the harmful interference brought by the sparse reward scheme is one of the focuses of this paper.

3. PROBLEM FORMULATION

In this section, a UAV navigation task from an initial to a target position is first formulated. The UAV will learn a policy by mapping internal state information to action sequences. Then the PAMDP modeling process with sparse reward is described in detail.

3.1. UAV navigation tasks

In this paper, UAV navigation tasks are divided into the direct navigation task [Figure 1] and the relay navigation task [Figure 2]. The distinction between these tasks lies in the presence of intermediate operations that the UAV needs to perform. As a result, the flight requirements for UAVs in relay navigation tasks are more demanding. Nevertheless, the simple task is also considered to verify the effectiveness and generality of the algorithm.

Typically, the position, direction, and motion of the UAV are determined by both the earth and the body coor-

dinate frame. The earth coordinate frame is used to describe the position and direction of the UAV, denoted as $\phi = [x, y, z, \theta_x, \theta_y, \theta_z]$. The linear and angular velocities of the UAV are denoted as $\psi = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z]$ in the body coordinate frame.

For simplicity, the UAV is assumed to fly at a fixed altitude, i.e., the motion of the UAV is constrained within the x-y plane. Moreover, the steering action of the UAV takes effect immediately because the momentum is ignored. As a result, the vector describing the motion information of the UAV is simplified to $\xi = [x, y, v, \theta]$, and the motion formula is expressed as follows

$$\begin{aligned} \theta_{t+1} &= \theta_t + \Delta\theta \\ v_{t+1} &= v_t + \Delta a \\ x_{t+1} &= x_t + v_{t+1}\cos(\theta_{t+1}) \\ y_{t+1} &= y_t + v_{t+1}\sin(\theta_{t+1}) \end{aligned} \tag{5}$$

where $\Delta\theta$ is the steering signal, Δa is the acceleration signal, v is the speed of the UAV, and θ is the direction angle between the UAV and the target.

3.2. State representation specification

State representation is essential for the agent to perceive the surrounding environment and reach the target position. In our setting, the state vector is reduced as much as possible to avoid interference from irrelevant information and speed up training. Specifically, the observable information of the UAV has three sources. The first is the internal state of UAVs, which is represented in terms of position $[x, y]$, velocity v, and direction θ. The second is the relationship between UAVs and the environment. Since obstacles are not considered, the number of actions the UAV performs is used to be the unique representation, which is recorded as n_{step}. Taking n_{step} as the state can encourage the UAV to complete the task with fewer steps. The last is the relationship information between the UAV and the target. Also, to simplify the representation, the distance between the UAV and the target is only used to describe this relationship, denoted as d_{target}. By combining the three kinds of information, the final form of the state vector becomes: $s = [x, y, v, \theta, d_{target}, n_{step}]$. For relay navigation tasks, UAVs need additional information about supplies. The location information of the supply can be directly obtained from the simulation environment. In this paper, the distance d_{supply} between the UAV and the supply is included as one of the state representations to fulfill this requirement. As a result, the state vector in the relay task is modified as follows: $s = [x, y, v, \theta, d_{target}, d_{supply}, n_{step}]$.

3.3. Parameterized action design

Considering that the flight altitude of the UAV has been set as a constant above, its movement in the vertical direction can be ignored. For the direct navigation task, the optional discrete actions of the UAV are defined in the discrete set $\mathbb{D} = \{k_1, k_2\}$. Where k_1 represents the "MOVE" behavior and k_2 represents the "TURN" behavior. In addition, the parameter set $\mathbb{C} = \{(c_1), (c_2)\}$ defines the continuous parameters corresponding to each action in $\mathbb{D}$, where c_1 represents the acceleration value and c_2 represents the rotation angle. When the UAV acts, it not only needs to select the discrete action but also needs to determine the corresponding parameter values. To sum up, the parameterized action space of UAV is represented as $\mathcal{H} = \{(k_1, (c_2)), (k_2, (c_2)\}$. The value range of c_1 and c_2 are scaled to $[-1 \sim 1]$. Since the UAV needs to perform intermediate operations for the relay navigation task, a new discrete action k_3 will be added to $\mathbb{D}$ to represent the "CATCH" behavior. It should be pointed out that action k_3 does not set continuous parameters. This is because we focus more on learning the navigation policy rather than the specific control process. When the UAV gets close enough to the supply, it performs the "CATCH" action, allowing the supply to move with itself. However, when the UAV is far away from the supply, it will not affect the supply when it performs the "CATCH" action. In this case, the UAV advances one step based on the current speed and angle. The primary goal of the UAV is to transport the supply to the target area.

3.4. Sparse reward function

Although sparse reward setting is simple and universal, different tasks need to set corresponding reward functions, which will bring different degrees of sparsity. The direct navigation task aims to get the UAV to reach the target area, which is a single-stage task with a relatively small reward sparsity. In the relay navigation task that contains supplies, an effective reward can be obtained only when the UAV finds supplies and carries them to the target area. This task includes relay operations which significantly increase the reward sparsity. Existing baselines are not effective at handling the relay task. This phenomenon and the solution are described in detail in Section 4.2. According to the above task requirements, the reward function of the direct navigation task is defined as

$$R = \begin{cases} 0, & if\ UAV\ in\ target\ area \\ -1, & otherwise \end{cases} \tag{6}$$

and the reward function of the relay navigation task is defined as

$$R = \begin{cases} 0, & if\ supply\ in\ target\ area \\ -1, & otherwise\ . \end{cases} \tag{7}$$

4. DEEP REINFORCEMENT LEARNING AGENT

To solve the navigation task presented in Section 3, the MPDQN algorithm is considered first, which has been proven effective in PAMDP. Besides, we introduce how to extend HER to solve the sparse reward problem in the relay task, followed by the implementation and training process of HER-MPDQN.

4.1. Multi-Pass Deep Q-Network (MP-DQN)

MP-DQN is an off-policy RL algorithm that deals with parameterized action space. In MP-DQN, the goal of the agent is to optimize the policy $\pi : S \rightarrow A$, which maximizes the long-term cumulative discounted rewards:

$$R_t = \sum_{i=t}^{T} \gamma^{t-i} r(s_t, k_t, x_{k_t}). \tag{8}$$

Due to the integration of DQN[29] and DDPG[30] algorithms, MP-DQN has a network architecture similar to the Actor-Critic architecture. In specific, the MPDQN agent has an actor-parameter network π_{θ_x} (similar to Actor) and a Q-value network Q_{θ_Q} (similar to Critic). Where θ_x and θ_Q are parameters of the network. The discrete action policy is implicitly learned when approximating the Q-value function.

For the actor-parameter network, the MP-DQN agent learns a deterministic mapping policy $\pi_{\theta_x}(s)$ from state to continuous parameters vector as follows

$$\mathcal{X}_k^t = \pi_{\theta_x}(s_t) \tag{9}$$

where $\mathcal{X}_k^t = [x_1^t, x_2^t, ..., x_K^t]$ contains all continuous parameters corresponding to the discrete actions k. In order to decouple discrete actions and irrelevant continuous parameters, the $\mathcal{X}_k^t$ becomes

$$\mathcal{X}_k^t = \begin{bmatrix} x_1^t & 0 & \cdots & 0 \\ 0 & x_2^t & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & x_K^t \end{bmatrix} = \begin{bmatrix} x_{e_1}^t \\ x_{e_2}^t \\ \vdots \\ x_{e_K}^t \end{bmatrix}$$

Each row of the matrix is fed into the Q-value network separately, thereby eliminating the influence of irrelevant parameters on the estimation of the Q-value function.

For the Q-value network, MP-DQN evaluates actions using an action-value function similar to DDPG. Since MPDQN needs to optimize both discrete actions and continuous parameters, the Bellman equation becomes

$$Q_{\theta_Q}(s_t, k_t, x_{k_t}) = \mathop{\mathbb{E}}_{r_t, s_{t+1}} \left[r_t + \gamma \max_{k \in [K]} \sup_{x_k \in \mathcal{X}_k} Q_{\theta_Q}(s_{t+1}, k, x_k) | s_t, k_t, x_{k_t} \right], \tag{10}$$

where k_t is the discrete action selected at time t, and x_{k_t} is the associated continuous parameter. To avoid taking supremum over continuous space $\mathcal{X}_k$, equation (10) is rewritten as:

$$Q_{\theta_Q}(s_t, k_t, x_{k_t}) = \mathop{\mathbb{E}}_{r_t, s_{t+1}} \left[r_t + \gamma \max_{k \in [K]} Q_{\theta_Q}(s_{t+1}, k, \pi_{\theta_x}(s_{t+1})) | s_t \right], \tag{11}$$

where $\pi_{\theta_x} : S \to \mathcal{X}_k$ represents the mapping relationship in equation (9). This means that approximating the Q-value function needs to fix θ_Q first and find θ_x such that

$$Q_{\theta_Q}(s, k, \pi_{\theta_x}(s)) \approx \sup_{x_k \in \mathcal{X}_k} Q_{\theta_Q}(s, k, x_k) \qquad for\ each\ k \in [K] \tag{12}$$

Then, similar to DQN, θ_Q is estimated by minimizing the mean-squared Bellman error. The loss function of the Q-value network is:

$$\mathcal{L}(\theta_Q) = \frac{1}{2} \left[y_t - Q_{\theta_Q}(s_t, k_t, \pi_{\theta_x}(s_t)) \right]^2, \tag{13}$$

where

$$y_t = r_t + \gamma \max_{k \in [K]} Q_{\theta'_Q}(s_{t+1}, k, \pi_{\theta'_x}(s_{t+1})). \tag{14}$$

The loss of the actor-parameter network is given by the negative sum of Q-values as

$$\mathcal{J}(\theta_x) = -\sum_{k=1}^{K} Q_{\theta_Q}(s_t, k, \pi_{\theta_x}(s_t)). \tag{15}$$

4.2. Hindsight experience replay

HER is another important baseline algorithm that we consider in our method. The core idea is to expand the experience by constructing the goal variable as shown in Figure3 (a). There are two ways to construct the goal variable:

1) Direct construction: The agent uses the target information feedback by the environment as the goal variable at step t, which is recorded as $desired\_goal_t$. Additionally, the location of itself at step t is recorded as $achieved\_goal_t$. In the relay navigation task, the traditional algorithm will always record the location of the supply as $achieved\_goal_t$. In our method, the information represented by $achieved\_goal_t$ changes with the state of the agent; see the end of this section for the specific process.
2) Replacement construction: Through direct construction, an experience can be obtained at step t, simplified as $(achieved\_goal_t, desired\_goal_t)$. The HER algorithm will randomly sample 4 items from the experience obtained from step $t+1$ to step T and then uses the $achieved\_goal$ as the goal variable to replace the $desired\_goal$ in the experience at step t. T is the maximum step that allows the agent to act in a single task.

The new goals constructed in the above way have the potential to be generalized to unseen real goals. Although the agent fails to achieve a given goal in the current episode, it still learns action sequences that may achieve different given goals in a future episode. Therefore, the original failure transition can be transformed into the virtual success transition by selecting a new goal from the state experienced to replace the initial goal.

However, HER tends to be less efficient in some relay tasks. As shown in Figure 3 (b), the goal of the agent is to deliver the block to the target area. It should be noted that the $achieved\_goal$ (i.e., the location of the block)

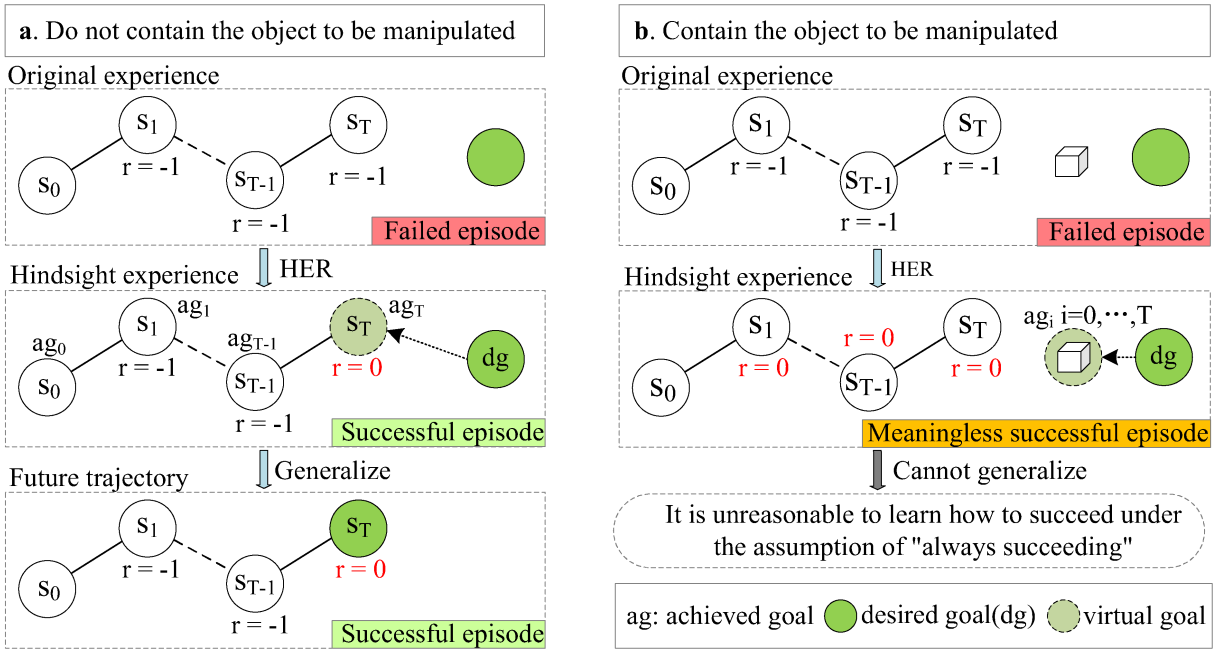

Figure 3. Illustration of positive reward sparsity for HER. (a) In the task that does not contain the manipulated object, achieved goal is directly affected by the behavior of the agent and constantly changes in each rollout. In this case, HER can generate valuable learning experiences. (b) For the task containing the manipulated object, achieved goal remains unchanged until the agent comes into contact with the object. In this case, all the experience generated by HER includes positive rewards but has no substantial help to the learning of the agent.

defined in HER has not changed when the agent is not in contact with the block. Any hindsight goal selected by HER is labeled as a "success episode". However, such a "successful episode" cannot bring meaningful guidance to the agent. This, in part, leads to another kind of sparsity, positive reward sparsity.

To solve this problem, we propose the goal-switch mechanism (GSM). The principle of the GSM aims to generate meaningful goal variables by dynamically assigning goals during experience expansion. In the relay navigation task, the conventional HER approach always designates the target area as the *desired_goal* and the supply point as the *achieved_goal*. GSM assists the UAV in determining the goal based on the current state: (1) Prior to acquiring the supply, the supply point is marked as the *desired_goal*, and the own position of the UAV is labeled as the *achieved_goal*. (2) After acquiring the supply, the target area is marked as the *desired_goal*, while the supply point remains as the *achieved_goal*. By assigning different goals, GSM enables the UAV to construct more effective goal variables and hindsight experiences, which helps mitigate reward sparsity. This simple idea provides an effective solution and is proved in Section 5.

4.3. HER-MPDQN

MP-DQN samples a fixed batch from an offline experience reply buffer to update the network. HER expands the original experience by goal replacement. On this basis, we further eliminate the hindsight experience without guidance significance. This means that our proposal can be effectively integrated with MP-DQN. As shown in Figure 4, the input vector of the neural network is extended in MP-DQN. In specific, the input vector of the actor-parameter network becomes

$$V_{\theta_x} = (s_t, g_t). \tag{16}$$

Correspondingly, the input vector of the Q-value network becomes

$$V_{\theta_Q} = (s_t, g_t, xe_{k_t}). \tag{17}$$

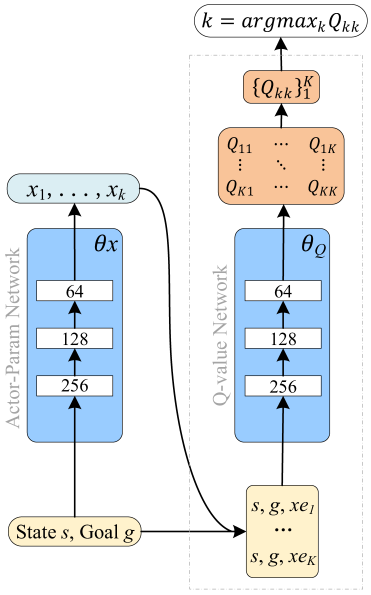

Figure 4. The network architecture of HER-MPDQN. For the symbols in the figure, s denotes the observed state, g represents the goal of the agent, θ_x and θ_Q refer to network parameters, and (256, 128, 64) indicates the number of neurons in the network. x_k is the continuous parameter corresponding to the kth discrete action, where $k = 1, 2, ..., K$. K is the total number of discrete actions. xe_k represents the expanded continuous parameter vector derived from x_k. Q_{kk} denotes the Q value associated with the kth discrete action. The selection of the discrete action k is determined based on the largest Q value.

In our method, the original experience is stored together with the hindsight experience in the replay buffer. A fixed batch of data is sampled from this buffer when the neural network needs to update. The training process in Algorithm 1 is generally divided into three steps: (1) The agent interacts with the environment to accumulate real experience; (2) Capture the original experience and hindsight experience of each time step and store all of them in the replay buffer; (3) Update the parameters of the network according to the length of each episode.

5. RESULTS

To evaluate the effectiveness of HER-MPDQN, a UAV is fully trained and tested in two environments. A relatively simple direct navigation environment and experimental results are first presented in 5.1. After that, a more complex relay navigation task is considered. The related experiments and result analysis are in 5.2.

5.1. The direct navigation task

To our knowledge, open-source benchmark environments for UAV navigation tasks with parameterized action spaces and sparse rewards are currently lacking. Therefore, we simulate a large-scale environment inspired by the existing simulation[1]. Figure 1 shows a top view of the navigation environment. The UAV can fly within four square kilometers ($2km \times 2km$). The same settings for the environment randomness are used to have a fair comparison. In each episode, the coordinate of the UAV and the target area are sampled from the uniform distribution of the entire environment. The default initial velocity of the UAV is 0. The goal of the UAV is to reach the target area in a limited number of steps. The optional actions are MOVE (acceleration) and TURN (angle). MOVE indicates that the UAV moves along the current direction with the given acceleration. TURN means that the UAV rotates at the given angle. The episode ends if the UAV reaches the target area (winning state) or the time limit exceeds.

[1]https://github.com/thomashirtz/gym-hybrid.

Algorithm 1: HER-MPDQN

Input: Minibatch size B, exploration parameter ϵ, soft update parameter τ, learning rate α_{θ_Q} and α_{θ_x}.
Initialize the weight of Q-value network and actor-parameter network.
Initialize target networks by hard update.
for $episode = 0, ..., max\_episode$ **do**
  Select an initial state s_0 and an initial goal g_0
  for $t = 0, ..., max\_step$ **do**
    Compute action parameters x_{k_t} by π_{θ_x}:
    $x_{k_t} = \pi_{\theta_x}(s_t||g_t)$
    Select action $a_t = (k_t, x_{k_t})$ by ϵ-greedy policy
    Recieve r_t, s_{t+1} and g_{t+1} by excute action a_t
  end
  for $t = 0, ..., episode\_length$ **do**
    Store the transition $(s_t, a_t, r_t, s_{t+1}, g_t)$ in R
    Sample additional goals G for current transition
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$
      Store the transition $(s_t, a_t, r', s_{t+1}, g')$ in R
    end
  end
  for $t = 0, ..., episode\_length * U$ **do**
    Sample a minibatch from R
    Compute the target y_i according to (5)
    Optimize parameters π_{θ_Q} and π_{θ_x}
    Update the weights of target networks by:
    $$\theta'_Q \leftarrow \tau * \theta_Q + (1-\tau) * \theta'_Q$$
    $$\theta'_x \leftarrow \tau * \theta_x + (1-\tau) * \theta'_x$$
  end
end

We chose Python 3.7 as our development language due to its simplicity, flexibility, and efficient development capabilities. For algorithm development and testing, we utilize the OpenAI Gym library, which is an open-source RL library that offers convenient tools for creating custom environments. In terms of hardware, we employ an AMD Ryzen 7 5800H processor and 16GB of RAM. This configuration is relatively new and provides sufficient computing resources to support the training and testing of the algorithms developed in this study.

In the context of sparse rewards, the agent ends the current episode either when the task completion or the agent goes out of bounds which results in higher rewards. Therefore, relying solely on "episode rewards" is inadequate to assess the learning effectiveness. To evaluate algorithm performance, we utilize the "success rate" as a metric, which directly reflects the number of times the agent completes the task. Also, a higher "success rate" implies a higher "reward". The "success rate" is defined as follows:

$$\mathcal{SR} = \frac{CT}{TT} \tag{18}$$

where $\mathcal{SR}$ means "success rate", CT is the number of times the agent completes the task, and TT is the number of times the agent performs the task.

We evaluated the performance of HER-MPDQN in the above environment. Besides, we also implement and test the following three baselines: HER-PDQN[31], MP-DQN,[21] and P-DQN[20]. Each algorithm uses the same network with hidden layer sizes (128,64). The step size of each environment is limited to 100, the size of

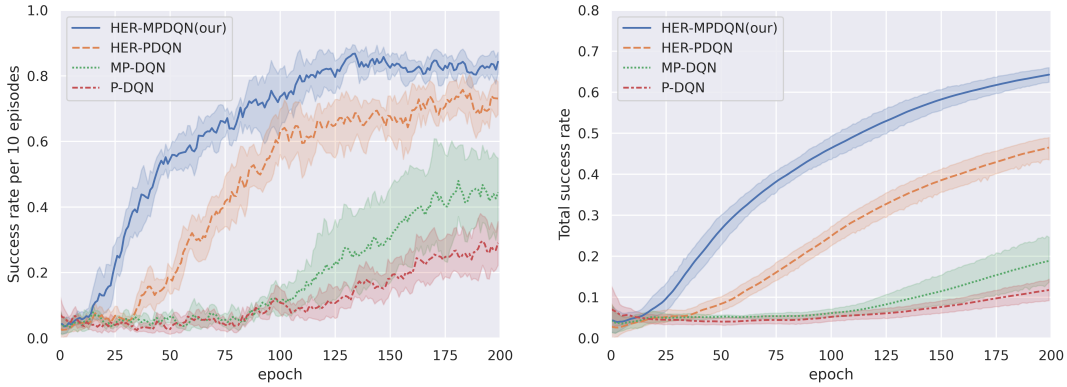

Figure 5. Comparisons of the results using HER-MPDQN and other baselines. The left figure shows the periodic calculation of the task completion rate of the last 10 episodes, and the right figure shows the total success rate during the entire training process. The shaded area is the variance in multiple experiments. Smaller shading indicates that the algorithm is less sensitive to random seeds.

Table 1. The average performance over 1000 episode evaluations

	The direct navigation task	
	Success rate	Mean reward
HER-MPDQN(our)	**0.810** ± **0.030**	**-40.095** ± **4.861**
HER-PDQN	0.725 ± 0.082	-43.141 ± 11.917
MP-DQN	0.412 ± 0.165	-70.504 ± 15.897
P-DQN	0.305 ± 0.102	-84.504 ± 13.125

the replay buffer R is 50, 000, the mini-batch size B is 128, and the update frequency U of the network is 40. The learning rate of the Q-value network α_{θ_Q} and actor-parameter network α_{θ_x} is 10^{-2} and 10^{-3} respectively. Adam optimizer with an exponential decay rate of ($\beta_1 = 0.9$, $\beta_2 = 0.999$) is used for optimization. All algorithms run seven independent experiments and train 2,000 episodes in each experiment.

Figure 5 provides the training process of the UAV in the direct navigation task. Table 1 shows the evaluation performance for each algorithm in 1000 episodes. The results show that HER-MPDQN has a faster convergence speed and higher average success rate. Compared to HER-PDQN, our method evaluates the Q-value more accurately and updates the actor-parameter network without bias. At the same time, the agent learns effective experience early by introducing HER, which brings stronger learning ability than the original MP-DQN. Since P-DQN has no additional mechanism to eliminate the influence of irrelevant parameters and deal with sparse rewards, its performance is not satisfactory. In addition, it can be seen that the learning curve of HER-MPDQN has a smaller shaded area. This means that HER-MPDQN has better robustness in various experiments.

5.2. The relay navigation task

Considering the need for UAVs to deliver supplies, a relay navigation environment is further simulated. Compared with the previous environment setting, the following changes are made: (1) Add a supply point that requires the UAV to perform relay operations. (2) Introduce a new discrete action CATCH to represent grabbing operations. As shown in Figure 2, the UAV must deliver supply to the target area in a limited number of steps. CATCH is valid only when the UAV is in contact with the supply. Each episode ends when the supply has been transported to the target area (winning state) or the time limit exceeds.

Due to the increased complexity, the hyperparameters for this task are tuned as follows. The hidden layer size of all networks is set to (256, 128, 64). The size of the replay buffer R is 150, 000. The update frequency U

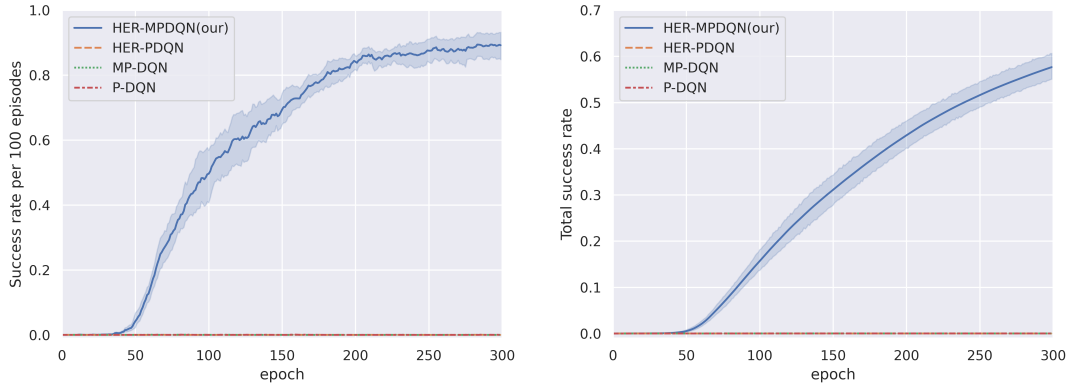

Figure 6. Comparisons of the results using HER-MPDQN and other baselines. The left figure shows the periodic calculation of the task completion rate of the last 100 episodes; other descriptions for evaluating algorithm performance are the same as Figure 5.

Table 2. The average performance over 1000 episodes evaluations

	The relay navigation task	
	Success rate	Mean reward
HER-MPDQN(our)	**0.885 ± 0.059**	**-57.758 ± 6.294**
HER-PDQN	0.000 ± 0.000	-00.000 ± 0.000
MP-DQN	0.000 ± 0.000	-00.000 ± 0.000
P-DQN	0.000 ± 0.000	-00.000 ± 0.000

changes according to the episode length and is limited to [1~10]. The learning rate of the Q-value network α_{θ_Q} and actor-parameter network α_{θ_x} is 10^{-3} and 10^{-5}, respectively. Each algorithm is trained with 30,000 episodes in each experiment.

Figure 6 shows the learning curve in the relay task. The evaluation results are given in Table2. It is obvious that HER-MPDQN is far superior to the other algorithms. On the one hand, more sparse rewards make it impossible for the UAV to complete tasks through random actions. Therefore, the MP-DQN and P-DQN without HER are challenging to learn. On the other hand, although many positive experiences are generated by introducing HER, they do not have guide significance. For this reason, HER-PDQN cannot learn policy effectively. By treating the relay task as a continuous multi-stage task, the goal is automatically allocated by HER-MPDQN according to the current state. Thus, the hindsight experience of different stages has the correct guiding significance. Figure 7 shows the flight trajectories of the trained UAV when performing specific relay navigation tasks. It can be seen that the UAV has learned the correct action policy. Not only that, HER-MPDQN exhibits good scalability in our experiments. The multi-goal relay navigation task can be completed by expanding the goal space. We leave this research for future work.

6. DISCUSSION

Existing baselines can learn effective policies in the direct navigation task but fail to handle the relay navigation task. In contrast, HER-MPDQN achieves satisfactory results in both simple and complex tasks. This means that HER-MPDQN is versatile in solving different types of navigation tasks. General-purpose agents are one of the critical directions of DRL research, which can avoid repeated algorithm design. HER-MPDQN has excellent advantages in tasks that are difficult to design reward functions and require flexible rescue strategies. However, the algorithm in environments with obstacles has yet to test, and the flying altitude of the UAV is fixed. These limitations make the transfer of simulation to reality more difficult. Future research can consider addressing the sparse reward problem in more realistic simulation conditions, which can narrow the gap

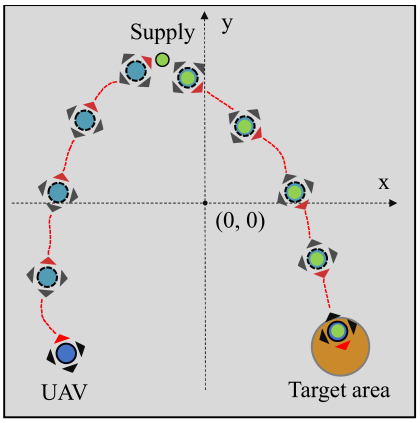

Figure 7. Illustration of flight trajectory map of the trained UAV in the relay navigation task. The red arrow represents the current direction of the UAV, and the red dotted line is the flight path.

between "sim-to-real".

7. CONCLUSIONS

In this paper, the HER-MPDQN algorithm is developed to address UAV navigation tasks with parametrized action space and sparse reward. In addition, a goal-switching method is proposed to correct meaningless hindsight experiences in the relay navigation task. The experiments show that HER-MPDQN outperforms baselines regarding training speed and converged value. Especially in the relay task, only the agent trained by HER-MPDQN learns effectively due to reasonable experience expansion. Further research could consider the energy consumption model of UAVs and test real UAVs in high-dimensional environments containing obstacles.

DECLARATIONS

Authors' contributions

Made contributions to the conception and design of the work, developed most associated code for the simulation environment and the reinforcement learning method, and performed data analysis and visualizations, manuscript writing, and related tasks: Feng S

Contributed to parts of the conception and design and performed the validation of the experimental design and related tasks: Li X

Contributed to the research supervision and guidance, provided environmental equipment, and performed analysis and transition of formal specifications, manuscript writing, and related tasks: Ren L

Participated in part of the experimental data analysis and visualizations and performed data collation and related tasks: Xu S

Availability of data and materials

This work is based on the Gym platform. All codes can be found at https://github.com/No-human-is-limited/HER-MPDQN. There are no additional data or materials associated with this study.

Financial support and sponsorship

This work was supported in part by the Open Project Program of Fujian Provincial Key Laboratory of Intelligent Identification and Control of Complex Dynamic System No:2022A0001; in part by the National Natural Science Foundation of China No:62203002; and in part by the Natural Science Foundation of Anhui Province No:2208085QF204.

Conflicts of interest
All authors declared that there are no conflicts of interest.

Ethical approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Copyright
© The Author(s) 2023.

REFERENCES

1. Mourtzis D. Simulation in the design and operation of manufacturing systems: state of the art and new trends. *Int J Prod Res* 2020;58:1927-49. DOI
2. Mourtzis D. Design and operation of production networks for mass personalization in the era of cloud technology. Amsterdam: Elsevier; 2021;1-393. DOI
3. Atif M, Ahmad R, Ahmad W, Zhao L, Rodrigues JJ. UAV-assisted wireless localization for search and rescue. *IEEE Syst J* 2021;15:3261-72. DOI
4. Zhao L, Liu Y, Peng Q, Zhao L. A dual aircraft maneuver formation controller for mav/uav based on the hybrid intelligent agent. *Drones* 2023;7:282. DOI
5. Mourtzis D, Angelopoulos J, Panopoulos N. Unmanned aerial vehicle (UAV) manipulation assisted by augmented reality (AR): the case of a drone. *IFAC-PapersOnLine* 2022;55:983-8. DOI
6. Walker O , Vanegas F , Gonzalez F ,et al. A deep reinforcement learning framework for UAV navigation in indoor environments. 2019 IEEE Aerospace Conference. 2019;1-14. DOI
7. Wang Q, Zhang A, Qi L. Three-dimensional path planning for UAV based on improved PSO algorithm. In: The 26th Chinese Control and Decision Conference (2014 CCDC). IEEE; 2014;3981-5. DOI
8. Yao P, Xie Z, Ren P. Optimal UAV route planning for coverage search of stationary target in river. *IEEE Trans Contr Syst Technol* 2017;27:822-9. DOI
9. Shin J, Bang H, Morlier J. UAV path planning under dynamic threats using an improved PSO algorithm. *Int J Aerospace Eng* 2020;2020:1-17. DOI
10. Wen C, Qin L, Zhu Q, Wang C, Li J. Three-dimensional indoor mobile mapping with fusion oftwo-dimensional laser scanner and RGB-D camera data. *IEEE Geosci Remote Sensing Lett* 2013;11:843-7. DOI
11. Mu B, Giamou M, Paull, et al. Information-based active SLAM via topological feature graphs. In: 2016 IEEE 55th Conference on decision and control (Cdc). IEEE; 2016. pp. 5583-90. DOI
12. Weiss S, Scaramuzza D, Siegwart R. Monocular-SLAM–based navigation for autonomous micro helicopters in GPS-denied environments. *J Field Robot* 2011;28:854-74. DOI
13. Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 2016;529:484-9. DOI
14. Levine S, Finn C, Darrell T, Abbeel P. End-to-end training of deep visuomotor policies. *J Mach Learn Res* 2016;17:1334-73. DOI
15. Levine S, Pastor P, Krizhevsky A, Ibarz J, Quillen D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Ind Robot* 2018;37:421-36. DOI
16. Yan C, Xiang X. A path planning algorithm for uav based on improved q-learning. In: 2018 2nd international conference on robotics and automation sciences (ICRAS). IEEE; 2018.;1-5. DOI
17. Bouhamed O, Ghazzai H, Besbes H, Massoud Y. Autonomous UAV navigation: A DDPG-based deep reinforcement learning approach. In: 2020 IEEE International Symposium on circuits and systems (ISCAS). IEEE; 2020. pp. 1-5. DOI
18. Masson W, Ranchod P, Konidaris G. Reinforcement learning with parameterized actions. In: Thirtieth AAAI Conference on Artificial Intelligence; 2016. DOI
19. Hausknecht M, Stone P. Deep reinforcement learning in parameterized action space. In: Proceedings of the International Conference on Learning Representations. 2016. DOI
20. Xiong J, Wang Q, Yang Z, et al. Parametrized deep q-networks learning: reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:181006394*. 2018. DOI
21. Bester CJ, James SD, Konidaris GD. Multi-pass q-networks for deep reinforcement learning with parameterised action spaces. *arXiv preprint arXiv:190504388*. 2019. DOI
22. Wang W, Luo X, Li Y, Xie S. Unmanned surface vessel obstacle avoidance with prior knowledge-based reward shaping. *Concurrency Computat Pract Exper* 2021;33:e6110. DOI
23. Okudo T, Yamada S. Subgoal-based reward shaping to improve efficiency in reinforcement learning. *IEEE Access*. 2021;9:97557-68. DOI
24. Burda Y, Edwards H, Storkey A, Klimov O. Exploration by random network distillation. *arXiv preprint arXiv:181012894*. 2018. DOI

25. Badia AP, Sprechmann P, Vitvitskyi A, et al. Never give up: learning directed exploration strategies. *arXiv preprint arXiv:200206038.* 2020. DOI
26. Andrychowicz M, Wolski F, Ray A, et al. Hindsight experience replay. *Adv Neural Inf Process Syst* 2017;30. DOI
27. Lanka S, Wu T. Archer: aggressive rewards to counter bias in hindsight experience replay. *arXiv preprint arXiv:180902070.* 2018. DOI
28. Schramm L, Deng Y, Granados E, Boularias A. USHER: unbiased sampling for hindsight experience replay. *arXiv preprint arXiv:220701115.* 2022. Available from: https://proceedings.mlr.press/v205/schramm23a/schramm23a.pdf [Last accessed on 15 Jun 2023]
29. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *nature.* 2015;518:529-33. DOI
30. Lillicrap TP, Hunt JJ, Pritzel A, et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:150902971* 2015. DOI
31. Liu C, Van Kampen EJ. HER-PDQN: a reinforcement learning approach for uav navigation with hybrid action spaces and sparse rewards. In: AIAA SCITECH 2022 Forum; 2022;0793. DOI