aerospace

MDPI

Article

A Reinforcement Learning Method Based on an Improved Sampling Mechanism for Unmanned Aerial Vehicle Penetration

Yue Wang *, Kexv Li, Xing Zhuang, Xinyu Liu and Hanyu Li

School of Mechatronical Engineering, Beijing Institute of Technology, Beijing 100081, China
* Correspondence: jackwy@bit.edu.cn

Abstract: The penetration of unmanned aerial vehicles (UAVs) is an important aspect of UAV games. In recent years, UAV penetration has generally been solved using artificial intelligence methods such as reinforcement learning. However, the high sample demand of the reinforcement learning method poses a significant challenge specifically in the context of UAV games. To improve the sample utilization in UAV penetration, this paper innovatively proposes an improved sampling mechanism called task completion division (TCD) and combines this method with the soft actor critic (SAC) algorithm to form the TCD-SAC algorithm. To compare the performance of the TCD-SAC algorithm with other related baseline algorithms, this study builds a dynamic environment, a UAV game, and conducts training and testing experiments in this environment. The results show that among all the algorithms, the TCD-SAC algorithm has the highest sample utilization rate and the best actual penetration results, and the algorithm has a good adaptability and robustness in dynamic environments.

Keywords: UAV penetration; reinforcement learning; sample utilization; task completion division

Citation: Wang, Y.; Li, K.; Zhuang, X.; Liu, X.; Li, H. A Reinforcement Learning Method Based on an Improved Sampling Mechanism for Unmanned Aerial Vehicle Penetration. *Aerospace* **2023**, *10*, 642. https://doi.org/10.3390/aerospace10070642

Academic Editor: Gokhan Inalhan

Received: 19 June 2023
Revised: 12 July 2023
Accepted: 14 July 2023
Published: 16 July 2023

1. Introduction

UAV penetration refers to the process in which UAVs change their original flight trajectory to avoid detection and interception by an enemy, thereby successfully infiltrating the enemy's defense. The ability to penetrate enemy defenses is crucial for UAV operations. However, the current research on autonomous UAV penetration is still limited, and there is a need to enhance the effectiveness of UAV penetration. This paper is dedicated to studying the issue of autonomous UAV penetration.

Previous studies on UAV penetration have primarily focused on constraints such as the position, speed, and overload of the re-entry point [1–3]. They have also examined performance measures like the heading error of the UAV [4] and analyzed the miss distance of interceptors to discuss UAV penetration strategies [5]. However, these traditional methods are primarily intended for planned trajectories before a launch and cannot be applied to real-time autonomous penetration after a launch.

In recent years, artificial intelligence methods represented by reinforcement learning (RL) have gradually been applied to UAV penetration. RL offers the advantage of enabling online autonomous penetration, which is not feasible with conventional approaches. The RL method iteratively updates and improves a pre-designed reward function, using a large number of samples to obtain the optimal policy for UAV penetration [6]. However, the significant sample requirements of RL pose time and cost challenges in the context of UAV penetration [7,8].

In order to improve the sample utilization in UAV penetration, this paper proposes a reinforcement learning method based on an improved sampling mechanism. The structure of this paper is as follows. Firstly, UAV penetration is modeled as a reinforcement learning problem, and the corresponding action space, state space, and reward function are designed. Then, according to the decomposability of the task in UAV penetration, a task completion

division (TCD) method is proposed to improve the sample utilization. Next, the TCD method is combined with the soft actor critic (SAC) algorithm, so as to obtain the TCD-SAC algorithm. Finally, the actual performance of the TCD-SAC algorithm in UAV penetration is analyzed through training and test experiments.

2. Related Works

The field studied in this paper belongs to UAV penetration, and algorithms in this field can be roughly divided into two categories. (1) Based on the swarm intelligence optimization algorithm. For example, Jing Luo [9] proposed a three-dimensional path planning algorithm based on improved holonic particle swarm optimization (IHPSO) to meet the requirements of safety, concealment, and timeliness of trajectory planning during the process of UAV penetration; Jinyu Fu [10] proposed a trajectory homotopy optimization framework for multiple UAVs, which solved the problem of dynamic penetration mission planning in complex environments; Zhe Zhang [11] proposed a penetration strategy based on an improved A-Star algorithm to solve the replanning problem of stealth UAVs in a three-dimensional complex dynamic environment. (2) Based on the reinforcement learning network. For example, Yuxie Luo [12] established an intelligent UAV model using the reinforcement learning network and built a reward function using the cooperative parameters of multiple UAVs to guide UAVs to conduct collaborative penetration; Yue Li [13] used reinforcement learning algorithms for training in four scenarios, frontal attack, escape, pursuit, and energy storage, thereby improving the intelligent decision-making level of air confrontation; Kaifang Wan [14] proposed a motion control method based on deep reinforcement learning (DRL), which provides additional flexibility for UAV penetration within the DRL framework; Liang Li [15] mainly focused on the winning region of three players in the reconnaissance penetration game, proposed an explicit policy method for analyzing and constructing barriers, and provided a complete solution by integrating the games of kind and degree.

In general, the swarm intelligence optimization method needs a long time period to update and iterate the final strategy, so it is not suitable for online real-time decision making. On the contrary, the reinforcement learning network has a strong data processing capability and the ability to make rapid decisions, which can meet the real-time online decision making of UAVs. Therefore, the reinforcement learning network is more suitable for the online autonomous penetration of UAVs.

Compared with the above references, the innovation of this paper lies in solving the problem of the high sample demand when RL is applied to UAV penetration. In addition, the related works often use the simplified three degrees of freedom particle model, while this study uses the six degrees of freedom aircraft model, so the results of the algorithm are closer to the application effect in the actual environment.

Specifically, this paper innovatively proposes an improved sampling mechanism called task completion division (TCD) and combines the TCD method with the soft actor critic (SAC) algorithm to form the TCD-SAC algorithm, designed to improve the sample utilization in UAV penetration. In addition, the feasibility of the algorithm is verified through a series of training and testing experiments.

3. Problem Description

3.1. UAV Penetration

UAV penetration refers to the penetration of drones into the opponent's defense during the game process. The application scenario of this paper is shown in Figure 1. First, the UAV is launched and continues to fly according to the predetermined trajectory to hit the target. However, the other party finds and tracks the UAV during the flight, launching an interceptor to intercept the UAV. When the UAV finds the interceptor, it maneuvers to penetrate the defense, thereby flying according to the new trajectory and finally hitting the target. The interceptor is a high-speed antimissile missile with guidance capability.

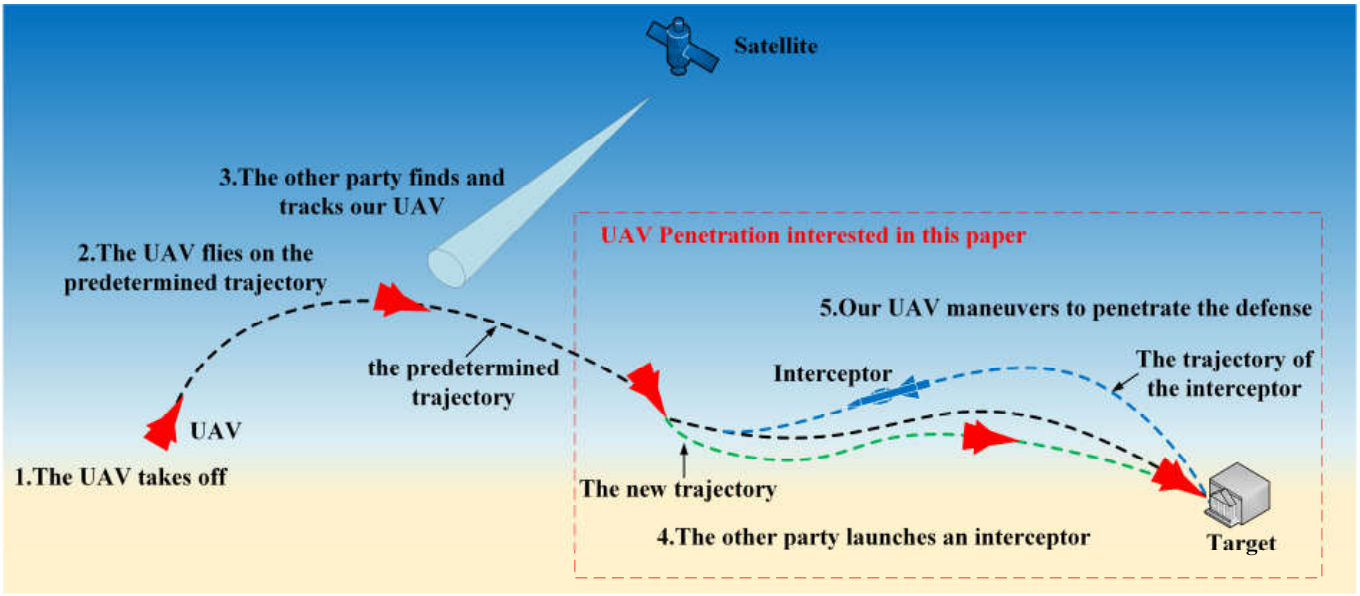


Figure 1. The application scenario for UAV penetration.

In the above process, the method in this paper will play a role in the process of UAV penetration, and its task is to guide the UAV to penetrate the defense of the interceptor and successfully hit the target. This method of UAV penetration will output maneuvering commands, so as to control the UAV to complete the task of autonomous penetration and a target hit.

3.2. Reinforcement Learning Description of UAV Penetration

The next state of the UAV penetration environment is related to the current state and the current action, so the process of UAV penetration can be modeled as a Markov decision process (MDP) [16]. Therefore, the problem of UAV penetration can be solved using the reinforcement learning algorithm.

Before research on the reinforcement learning algorithm is carried out, a six degrees of freedom model of UAV should be built, so as to conduct subsequent training and test experiments. The UAV model built in this study is a hypersonic glider based on real parameters [17]. To facilitate calling, this study converts the model into a dynamic link library, which can also reduce the CPU usage and improve the computational efficiency to a certain extent.

For the control system of UAV, the design methods of the control system mainly include an attitude control method and overload control method [18]. The hypersonic glider built in this study is a complex aircraft with a large lift/drag ratio, which means that the attitude control method will have a better stability margin and reliability [19]. Therefore, this study adopts the attitude control method and selects the angles of attack and bank as the attitude control command to control the UAV.

The relationship between UAV penetration and reinforcement learning is shown in Figure 2. The reinforcement learning algorithm sends the attitude control command to the UAV, and the UAV receives and executes the command to penetrate the interceptor, which intercepts the UAV at the same time. The adversarial environment composed of the UAV and interceptor outputs the current state and reward. The reinforcement learning algorithm receives the current state and reward and updates the internal parameters of the algorithm accordingly to better output the next action command.

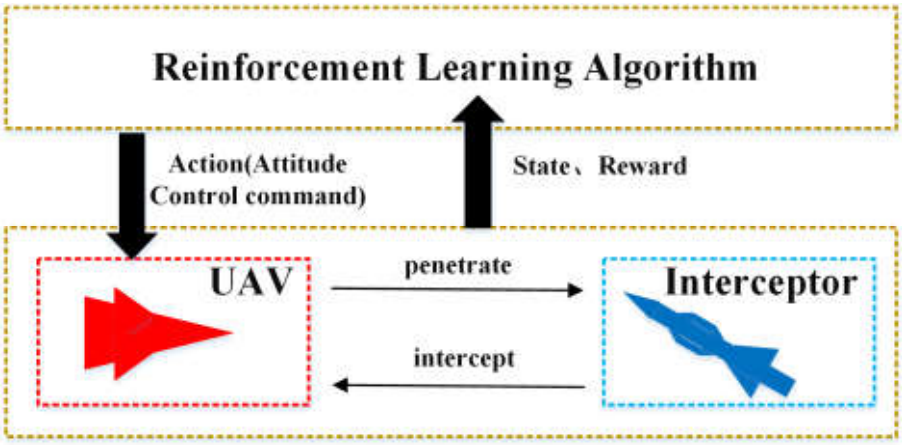


Figure 2. The relationship between UAV penetration and reinforcement learning.

During the MDP process of UAV penetration, the action vector a_t, state vector s_t, and reward function r_t of the penetration environment need to be determined. The specific design method is as follows.

3.2.1. State and Action Specification

The input of the control system is the attitude control command, so the angles of attack α and bank γ_V are selected as the action vector of the reinforcement learning network. Therefore, the action vector a_t can be expressed as Formula (1).

$$a_t = (\alpha, \gamma_V) \tag{1}$$

In addition, to ensure flight stability, the ranges of attack angle α and bank angle γ_V are $[-20^\circ, 20^\circ]$ and $[-60^\circ, 60^\circ]$, respectively.

The variables related to the observation state are shown in Figure 3. The red icon represents our UAV, the blue icon represents the opponent's interceptor, the yellow icon represents the target to be hit, and the red dashed line represents the trajectory of our UAV. This paper defines the relative distance between the UAV and interceptor as R_1, the line-of-sight angle of the UAV relative to the interceptor as q_1, and the relative distance between the UAV and target as R_2.

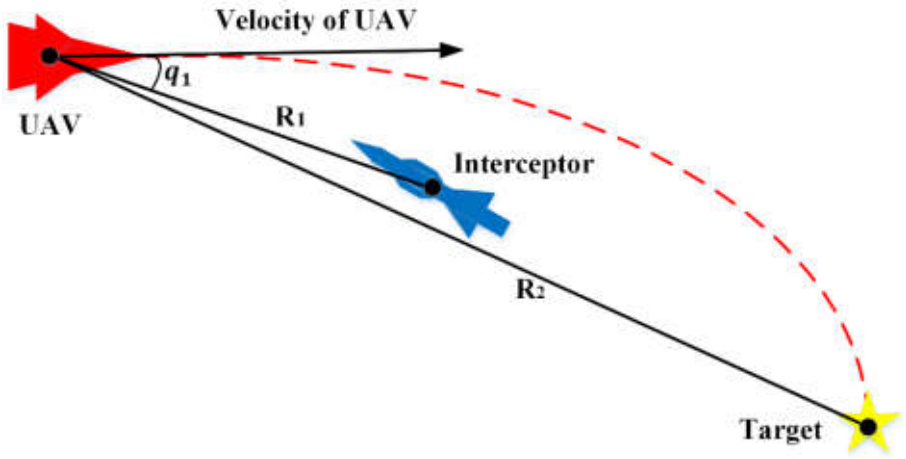


Figure 3. The variables related to the observation state.

In order to complete the task of autonomous penetration and hitting the target, the algorithm needs to obtain the relative distance and relative angle of the UAV relative to the interceptor and target. That is to say, the state information to be obtained includes the relative distance between the UAV and interceptor (R_1), the line-of-sight angle of the UAV relative to the interceptor (q_1), and the relative distance between the UAV and target (R_2), so a three-dimensional state space is built, and its corresponding state vector s_t is shown as Formula (2).

$$s_t = (R_1, q_1, R_2) \tag{2}$$

In actual application, the state vector s_t will be used as the input of the reinforcement learning network, and the network will output the corresponding action vector a_t based on the state vector s_t.

3.2.2. Reward Shaping

The reward function plays a vital role in reinforcement learning, and it will be used as the optimal target function for subsequent training. The environment for UAV penetration has high-dimensional and continuous characteristics, so the exploration space of the reinforcement learning algorithm is huge [20]. If there is only a main reward, it is difficult to obtain positive samples [21,22] (successful samples of autonomous penetration and a target hit). Therefore, it is considered to add a process reward in reward shaping to solve the problem of convergence difficulty.

The primary objective of this algorithm is to guide the UAV to penetrate the defense system and successfully strike the target. Consequently, the design of the reward function focuses on two key aspects:

(a). Guiding the UAV through interception defenses

Guiding the UAV through interception defenses

When the UAV encounters interception measures, a greater relative distance R_1 between the two aircrafts increases the likelihood of evading the interceptor and achieving a higher success rate of penetration. Therefore, the reward function should assign a correspondingly higher value. The corresponding reward function is shown in Formula (3).

$$r_{\mathrm{t}} = a_1 e^{b_1 \times R_1} \tag{3}$$

where $a_1 > 0,\ b_1 > 0$, and r_{t} is the increasing function of R_1.

Additionally, a larger line-of-sight angle q_1 between the UAV and the interceptor also enhances the success rate of penetration. Thus, the reward function should give a more significant weight to such scenarios. The corresponding reward function is shown in Formula (4).

$$r_{\mathrm{t}} = a_2 \arctan(b_2 \times q_1) \tag{4}$$

where $a_2 > 0$, $b_2 > 0$, and r_{t} is the increasing function of q_1.

(b). Guiding the UAV to hit the target after penetration

Once the successful penetration through the defense system is confirmed, the algorithm must direct the UAV towards the designated target to accomplish the mission of striking the target. During this phase, a shorter relative distance R_2 between the UAV and the target corresponds to a higher success rate of hitting the target. Therefore, the reward function should assign a greater value to scenarios where the UAV is in closer proximity to the target. The corresponding reward function is shown in Formula (5).

$$r_{\mathrm{t}} = a_3 e^{b_3 \times R_2} \tag{5}$$

where $a_3 > 0$, $b_3 < 0$, and r_{t} is the decreasing function of R_2.

According to the above analysis, the pseudo code of the reward function is outlined in Function 1.

Function 1: Reward Function of UAV Penetration.

$r_1 = r_2 = r_3 = r_4 = r_5 = r_6 = r_7 = 0$
$r_t = r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + r_7$
If the UAV is intercepted:
$r_1 = a_1 e^{b_1 \times R_1}$, where $a_1 > 0,\ b_1 > 0$, r_1 is the increasing function of R_1
$r_2 = a_2 \arctan(b_2 \times q_1)$, where $a_2 > 0$, $b_2 > 0$, r_2 is the increasing function of q_1
If the UAV penetrates successfully:
$r_3 = c_1$, where $c_1 > 0$, r_3 is the instant reward for successful penetration
If the UAV fails to penetrate:
$r_4 = c_2$, where $c_2 < 0$, r_4 is the immediate penalty for failed penetration
If the UAV successfully penetrates and before landing:
$r_5 = a_3 e^{b_3 \times R_2}$, where $a_3 > 0$, $b_3 < 0$, r_5 is the decreasing function of R_2
If the UAV lands and hits the target:
$r_6 = c_3$, where $c_3 > 0$, r_6 is the instant reward for hitting the targetIf the UAV lands and misses the target:
$r_7 = c_4$, where $c_4 < 0$, r_7 is the immediate penalty for missing the target
$r_t = r_1 + r_2 + r_3 + r_4 + r_5 + r_6 + r_7$

To facilitate the convergence of the algorithm, this paper sets the value of the reward within the range of $[-1, 1]$. The specific values for the theoretical parameters mentioned above are assigned based on the actual performance parameters of the UAV model constructed.

This paper divides the UAV's flight phase into several distinct phases. The phase in which the UAV is not intercepted by the interceptor is referred to as Phase 1. Phase 2 represents the phase in which the UAV is being intercepted. Phase 3 corresponds to the phase in which the UAV successfully penetrates the interceptor. Phase 4 denotes the phase in which the UAV fails to penetrate. Phase 5 represents the phase after successful penetration and before landing. Phase 6 represents the phase of landing and successfully hitting the target, while Phase 7 refers to the phase of landing but missing the target.

Based on these phases, the specific form of the reward function is defined as shown in Formula (6).

$$r_t = \begin{cases} 0 & , phase\ 1 \\ 0.23 \times e^{0.0023 R_1} + 0.17 \times arctan(q_1) & , phase\ 2 \\ -1 & , phase\ 3 \\ 1 & , phase\ 4 \\ 0.35 \times e^{0.0012 R_2} & , phase\ 5 \\ 1 & , phase\ 6 \\ -1 & , phase\ 7 \end{cases} \tag{6}$$

Assuming that the warhead of the interceptor is an antipersonnel warhead, then the judgment condition for failed penetration is as follows: the relative distance between the UAV and interceptor R_1 is less than 300 m; otherwise, it is considered a successful penetration. The judgment condition for hitting the target is as follows: the relative distance between the UAV and target R_2 during the landing time is less than 10 m; otherwise, it is considered to have missed the target.

4. TCD-SAC Algorithm

4.1. SAC Algorithm

Both the action space and state space of the UAV are continuous, so the problem of UAV penetration should be solved with a continuous control algorithm. As shown in Figure 4, the continuous control algorithms in reinforcement learning are mainly divided into three categories: the proximal policy optimization (PPO) algorithm [23], the deterministic policy gradient (DDPG) algorithm [24], and the soft actor critic (SAC) algorithm [25]. The PPO algorithm is an on-policy algorithm, which faces the serious problem of sample efficiency, requiring a large number of samples to learn; the DDPG algorithm is an off-policy algorithm with a deterministic policy, which means that only the best action is considered in each state, so its exploration ability is poor; the SAC algorithm is an off-policy algorithm with a

stochastic policy, which has a higher sampling efficiency than the PPO algorithm and has a stronger exploration ability than the DDPG algorithm.

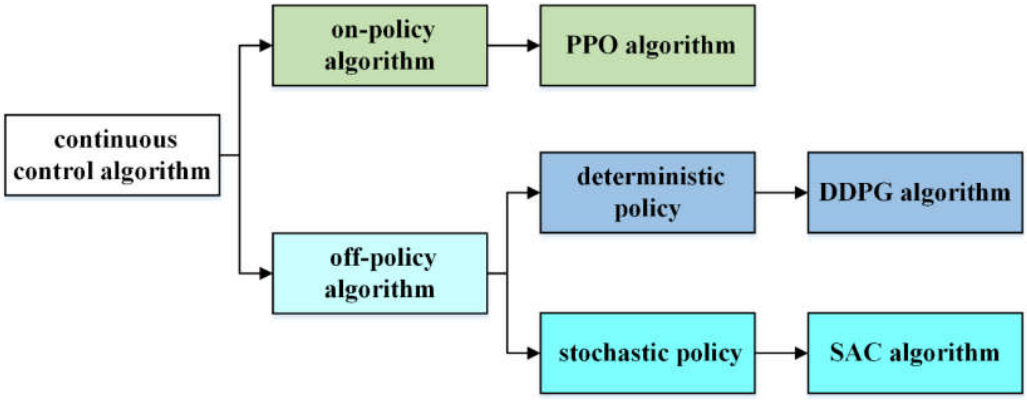


Figure 4. Continuous control algorithm in reinforcement learning.

There are many action combinations that can realize UAV penetration. Therefore, the algorithm for UAV penetration should use a stochastic policy to explore as many combinations as possible. At the same time, the cost of training for UAV penetration is high, so it is better to realize convergence with less sampling. Based on the above considerations, the SAC algorithm was initially selected for UAV penetration, with lower sample requirements and a stronger exploration ability.

The framework of the SAC algorithm is shown in Figure 5. The SAC algorithm consists of a Policy network, a Value network, and two Q networks. (1) The Policy network generates the optimal action by observing the current state of the environment. Therefore, the input of the Policy network is the state, and the output is the probability distribution of action, from which an action can be sampled. (2) The Value network is a value estimation network of state, which takes the current state as the input and outputs an evaluation value for the current state. (3) The Q network is a value estimation network of a state–action pair, which takes the state–action pair as the input and outputs an evaluation value for the current pair. The specific theoretical derivation of the SAC algorithm is provided in Appendix A.

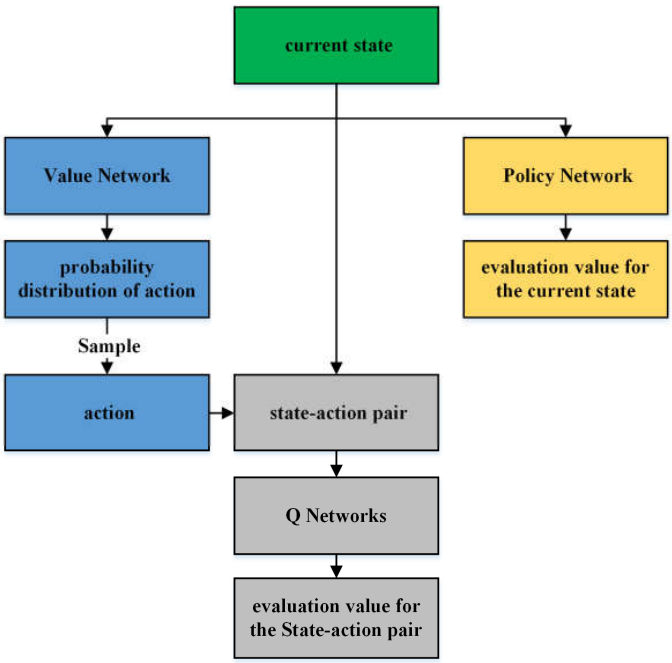


Figure 5. Framework of the SAC algorithm.

4.2. Task Completion Division

Facing a huge exploration space of UAV penetration, the number of samples required for network convergence is high. Although the sample demand of the SAC algorithm is relatively small, the actual number of samples required is still huge [25]. To improve the sample utilization of UAV penetration, this paper proposes an improved sampling method, which will improve the sampling efficiency by dividing task completion. This paper defines the improved method as task completion division (TCD).

Specifically, the task of the UAV in this paper is autonomous penetration and a target hit, so its task can be preliminarily divided into two subtasks: a (1) penetration subtask and (2) hit subtask. For example, there is a sample that successfully penetrates but fails to hit the target; that is to say, the penetration subtask has been completed, but the hit subtask has not been completed. In the above example, the sample only completed part of the total task, so it will be regarded as worthless [26–28]. Although this sample did not hit the target, its successful action combination of penetration is of learning value. Therefore, if the successful part of this sample can be used for learning, it will be useful to reduce the sample demand.

During the training process of UAV penetration, the UAV will continuously interact with the environment to collect samples. In these samples, each episode corresponds to a trajectory S, which is based on the total task m. The trajectory S consists of all states from the start to the end of the corresponding episode, which is shown as Formula (7).

$$S = (s_1, \ldots, s_T) \tag{7}$$

where s_1 is the state at the start time, s_T is the state at the end time, and T is the total time spent in the episode.

Define the subtask set completed by S as M, and all subtasks included by M are shown in Formula (8).

$$M = (m_1, \ldots, m_T) \tag{8}$$

where m_1 is the subtask completed in state s_1 and m_T is the subtask completed in state s_T. If the total task m can be completed after all subtasks in M are combined, then the corresponding trajectory S is considered a positive sample; otherwise, S is a negative sample.

For a negative sample, the corresponding trajectory S does not provide any useful information. However, although S did not complete the total task m, it completed some subtasks. If these subtasks are used to replace the original total task m, and the value of reward in the trajectory S is recalculated, some useful information can be obtained. In the method of TCD, this trajectory S can select another task m_t from M, and play back again, so as to learn from the successful part of the negative sample.

Based on the original task m, the original transition T_{ori} is expressed as Formula (9).

$$T_{ori} = (s_t||m, a_t, r_t, s_{t+1}||m) \tag{9}$$

where $s_t||m$ represents a concatenation operation of the current state s_t and original task m, $s_{t+1}||m$ represents a concatenation operation of the next state s_{t+1} and original task m, a_t is the action at the current time, and r_t is the reward of the original task m at the current time.

Based on the new task m_t, the new transition T_{new} is expressed as Formula (10).

$$T_{new} = \left(s_t||m_t, a_t, r'_t, s_{t+1}||m_t\right) \tag{10}$$

where $m_t \epsilon M$, $s_t||m_t$ represents a concatenation operation of the current state s_t and new task m_t, $s_{t+1}||m$ represents a concatenation operation of the next state s_{t+1} and new task m_t, and r'_t is the reward of the new task m_t at the current time.

As for the division of m_t, it is judged according to the states of S. When UAV penetration is successful, it is considered to have completed penetration subtask m_{pen}, so the corresponding subtask m_t is shown as Formula (11).

$$m_t = m_{pen} \tag{11}$$

where the condition for completing penetration subtask m_{pen} is that the relative distance between the UAV and interceptor is always greater than 300 m and the UAV has passed the interceptor.

When the UAV hits the target successfully, it is considered to have completed hit subtask m_{hit}, so the corresponding subtask m_t is shown as Formula (12).

$$m_t = m_{hit} \tag{12}$$

where the condition for completing hit subtask m_{hit} is that the relative distance between the UAV and target is less than 10 m.

When both m_{pen} and m_{hit} are successfully completed, it indicates that the total task m has been completed, so the corresponding subtask m_t is shown as Formula (13).

$$m_t = m \tag{13}$$

In this case, many successful subtasks will be used as positive samples, and the number of positive samples will increase many times, thus increasing the sample utilization. A more formal description of TCD is shown in Algorithm 1.

Algorithm 1: task completion division (TCD) algorithm

Given:
a reinforcement learning algorithm $\mathbb{A}$ ▷ e.g., SAC, DDPG, PPO
a reward function r: $\mathcal{S} \times \mathcal{A} \times \mathcal{M} \to \mathbb{R}$.
Initialize $\mathbb{A}$
Initialize replay buffer $\mathcal{D}$
for episode = 1, K **do**
  Set a task m and sample an initial state s_0.
  For $t = 0, T-1$ **do**
    Sample an action a_t using the behavioral policy from $\mathbb{A}$:
      $a_t \leftarrow \pi_b(s_t||m)$
    Execute the action a_t and observe the next state s_{t+1}
  end for
  for $t = 0, T-1$ **do**
      $r_t := r(s_t, a_t, m)$
    Store the transition $(s_t||m, a_t, r_t, s_{t+1}||m)$ in $\mathcal{D}$ ▷ standard experience replay
    for $m_t \epsilon M$ **do**
      $r_t^{'} := r(s_t, a_t, m_t)$
      Store the transition $\left(s_t \middle|\middle| m_i, a_t, r_t^{'}, s_{t+1} \middle|\middle| m_i\right)$ in $\mathcal{D}$ ▷ TCD
    end for
  end for
  for $t = 1, L$ **do**
    Sample a minibatch B from the replay buffer $\mathcal{D}$
    Perform one step of optimization using $\mathbb{A}$ and minibatch B
  end for
end for

4.3. TCD-SAC Algorithm

This paper combines the above TCD algorithm with the SAC algorithm and finally obtains the TCD-SAC algorithm, whose framework is shown in Figure 6. The TCD-SAC algorithm samples from the replay buffer and outputs a corresponding action based on the current state. This action controls the UAV to engage in game confrontation with the interceptor, and the transition generated during the confrontation process is placed in the replay buffer. The above process is continuously cycled until the algorithm converges.

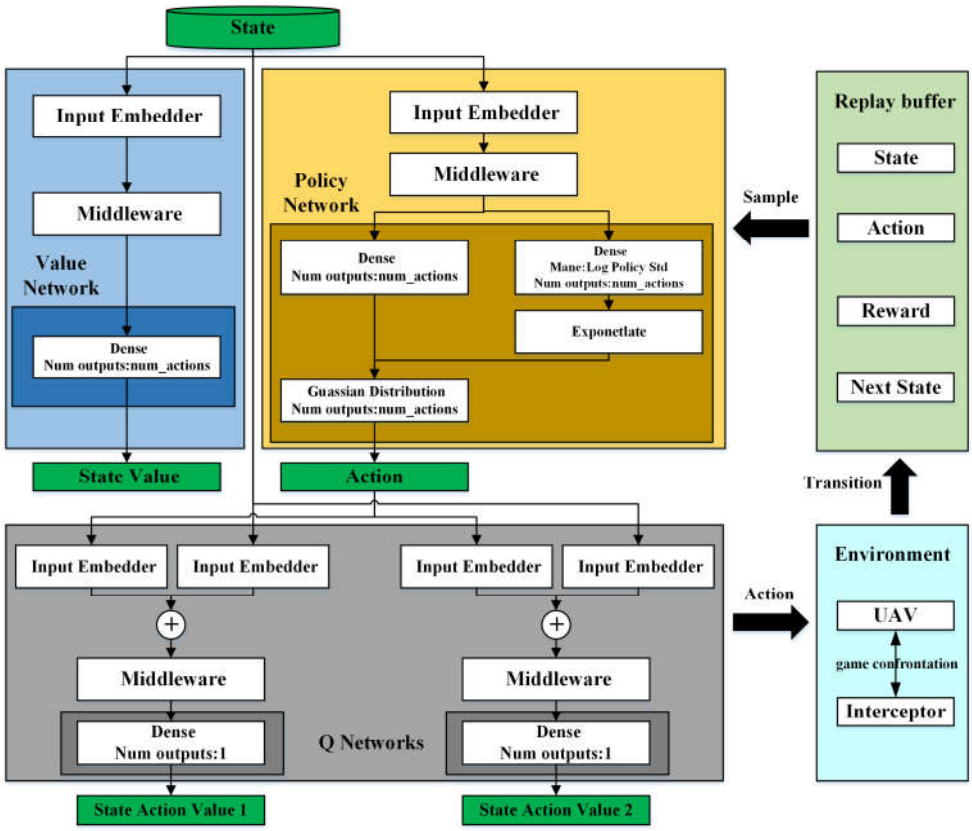


Figure 6. Framework of TCD-SAC algorithm.

The pseudo code of the TCD-SAC algorithm is shown in Algorithm 2. First, input initial parameter vectors θ_1, θ_2, and ϕ. Then, initialize target network weights $\bar{\theta}_1$ and $\bar{\theta}_2$, and initialize an empty replay pool $\mathcal{D}$. Next, in each time step t, sample action a_t from the policy π_ϕ, sample the transition $(s_t||m, a_t, r_t, s_{t+1}||m)$ from the environment, and store it in the replay pool $\mathcal{D}$. For the subtask $m_t \epsilon M$, store the additional transition $(s_t||m_t, a_t, r'_t, s_{t+1}||m_t)$ in the replay pool $\mathcal{D}$, so as to increase the number of valuable transitions [29–31]. Then, in each gradient step, update the Q-function parameters θ_1 and θ_2, update policy weights ϕ, adjust temperature α, and update target network weights $\bar{\theta}_1$ and $\bar{\theta}_2$, so as to output the optimized parameters θ_1, θ_2, and ϕ.

Because of the addition of the TCD algorithm, the TCD-SAC algorithm can learn from failure. For example, when the UAV fails to penetrate the defense in a certain episode because it is too close to the interceptor, a UAV with conventional algorithms cannot learn any useful information from this failed sample [32,33]. Instead, based on the experience of the previous failure, a UAV with the TCD-SAC algorithm will distance itself from the interceptor in the next episode, thereby increasing the probability of successful penetration [34–36].

Algorithm 2: task completion division–soft actor critic (TCD-SAC) algorithm

Input: θ_1, θ_2, ϕ
$\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
$\mathcal{D} \leftarrow \varnothing$
for each iteration **do**
for each environment step **do**
$a_t \sim \pi_\phi(a_t|s_t)$
$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
$r_t := r(s_t, a_t, m)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{((s_t||m, a_t, r_t, s_{t+1}||m))\}$
for $m_t \epsilon M$ **do**
$r' := r(s_t, a_t, m_t)$
$\mathcal{D} \leftarrow \mathcal{D} \cup \{((s_t||m_t, a_t, r'_t, s_{t+1}||m_t))\}$
end for
end for
for each gradient step **do**
$\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ *for* $i \in \{1, 2\}$
$\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
$\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$
$\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ *for* $i \in \{1, 2\}$
end for
end for
Output: θ_1, θ_2, ϕ

Theoretically, the TCD-SAC algorithm is able to improve the efficiency of sampling and speed up the convergence of the network. However, the above is only a theoretical derivation. In order to verify the real performance of the algorithm, experiments should be carried out.

(1) To verify the performance of the TCD-SAC algorithm, it should be compared with the original SAC algorithm.

(2) To verify the superiority of the SAC algorithm in UAV penetration, it should be compared with the DDPG algorithm and PPO algorithm (the other two excellent algorithms mentioned above). Of course, the TCD-DDPG algorithm and TCD-PPO algorithm should be compared with the TCD-SAC algorithm, where the TCD-DDPG algorithm is a combination of the DDPG and TCD algorithms, and the TCD-PPO algorithm is a combination of the PPO and TCD algorithms.

Therefore, the TCD-SAC algorithm, TCD-DDPG algorithm, TCD-PPO algorithm, SAC algorithm, DDPG algorithm, and PPO algorithm need to be compared through subsequent experiments, and the experimental results will be presented and discussed in detail in Section 5.

5. Environmental Results and Discussion

5.1. Training and Testing Environment

For the convenience of observation, this study builds a complete visual simulation environment based on the 6-DOF UAV model. The parameters of the UAV model are based on real data, the visual part is written in the QT language, and the algorithm is written in the Python language; the algorithm calls the 6-DOF UAV model through the User Datagram Protocol.

As for the 6-DOF UAV model built in this study, its range is approximately 4000 to 6000 km, its maximum boost height is approximately 120~180 km in the boost section, its glide height is approximately 30~40 km in the gliding section, and its gliding speed is approximately 4~6 Ma. The specific physical parameters of the UAV are shown in Table 1.

Table 1. Physical parameters of the UAV.

Performance	Value
range	4000~6000 km
maximum boost height	120~180 km
glide height	30~40 km
gliding speed	4~6 Ma

As for the reinforcement learning algorithm for UAV penetration, it is in the software environment of Python 3.6 and the Pytorch framework. The actual training and test process is shown in Figure 7. The specific parameters of each algorithm are shown in Table 2.

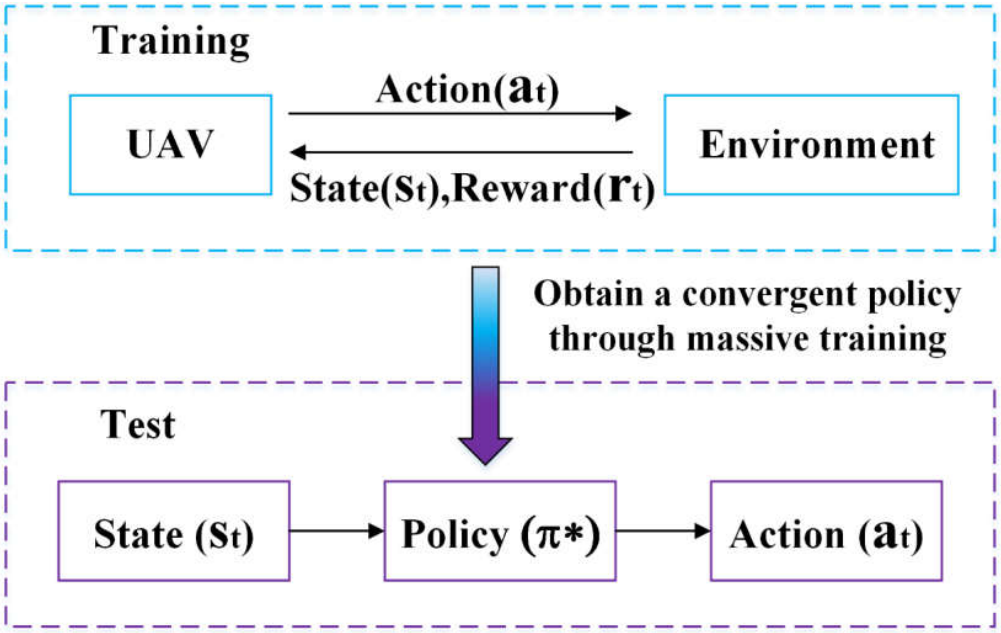


Figure 7. The actual training and test process of the algorithm.

Table 2. Parameters of each algorithm.

Algorithm	Network Dimension Number	Network Layer Number	Replay Buffer	Batch Size	Update Times	Discount Factor
SAC	128	4	2^{17}	128	1024	0.998
DDPG	256	4	2^{17}	256	512	0.997
PPO	128	8	2^{17}	128	1024	0.994
TCD-SAC	128	4	2^{17}	128	1024	0.995
TCD-DDPG	256	4	2^{17}	256	512	0.991
TCD-PPO	128	8	2^{17}	128	1024	0.997

The process of training: When the UAV detects the interceptor in the environment, the perception unit of the UAV will obtain the state vector $s_t = (R_1, q_1, R_2)$ and transmit it to the algorithm. Inside the algorithm, the corresponding action vector $a_t = (\alpha, \gamma_V)$ will be given according to s_t, and the reward function r_t will be iteratively updated, whose value is obtained from Formula (3). With the progress of massive training, the reward function gradually converges to the optimal value, thus obtaining a policy π^*, which can stably output a real-time action command (α, γ_V).

The process of testing: The weight parameters of the algorithm will be fixed after convergence. At this time, when the state vector s_t is observed again by the perception unit, the stable policy π^* will output a continuous action command (α, γ_V) to control the

flight. Under the guidance of policy π^*, the UAV can achieve autonomous evasion and penetration and finally hit the target.

5.2. Performance in Training Experiment

The performances of all algorithms (SAC, DDPG, PPO, TCD-SAC, TCD-DDPG, and TCD-PPO) are trained in a dynamic environment. The operating system is Ubuntu 18.04, and the GPU is a GeForce RTX 3090. The maximum number of episodes is 10,000, and the maximum time step for each episode is 2000. Under these conditions, it takes 6~8 h to perform a training session with 10,000 episodes. The launch point and target point of the UAV are randomly initialized in each episode, and the performance of each algorithm is as shown in Figure 8. The abscissa in Figure 8 is the episodes, which means the number of training rounds, while the ordinate is the average reward, which means the average value of all groups of rewards. As for each algorithm, this study sets the value of the seed to vary from 0 to 9, corresponding to 10 rounds of training, so the average reward of an ordinate is the average value of 10 groups of rewards.

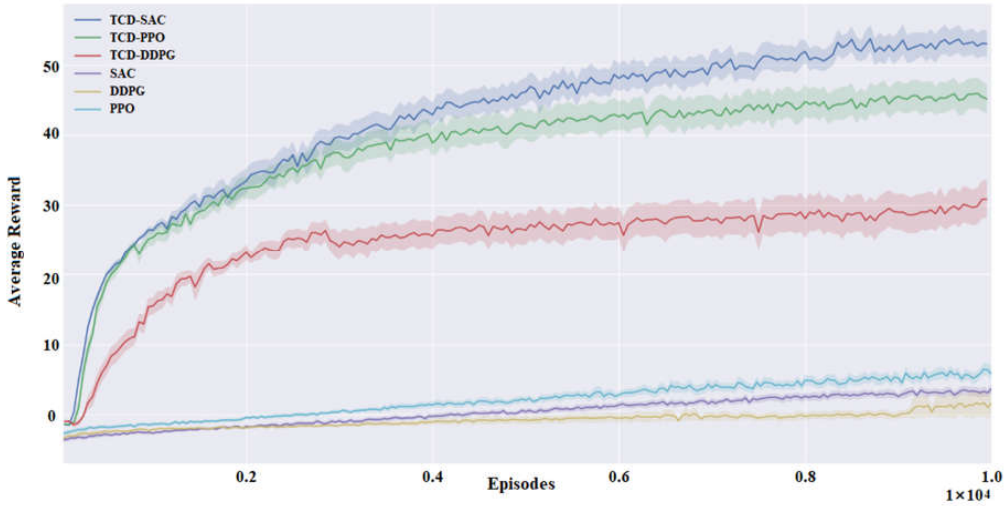


Figure 8. The average reward for each algorithm.

From Figure 8, the performance of each algorithm is analyzed as follows.

(1) From the convergence speed and convergence result: When the SAC algorithm is applied alone, the network cannot converge after 10,000 episodes, and the value of the average reward is low, indicating that the original SAC algorithm has poor performance in UAV penetration. However, when the TCD-SAC algorithm is applied, the algorithm converges at about 8000 episodes, and its average reward value corresponding to convergence is the highest of all algorithms, indicating that the TCD-SAC algorithm has the fastest convergence speed and the best convergence result.

(2) From the perspective of sample utilization: When the SAC, PPO, or DDPG algorithm is used alone, the network does not converge after 10,000 episodes, which is because their sample utilization is too low. However, when the above three algorithms are combined with the TCD method to obtain the TCD-SAC, TCD-PPO, and TCD-DDPG algorithms, they can all converge, which shows that the TCD method effectively improves the sample utilization. Moreover, the TCD-SAC algorithm with the fastest convergence speed shows that it has the highest sample utilization.

(3) From the superiority of the SAC algorithm in UAV penetration: When the TCD-DDPG algorithm is used, its network converges at about 4000 episodes, and its convergence result is lower than that of the other two algorithms (TCD-SAC, TCD-PPO). This is because the algorithm with a deterministic policy selects only one deterministic action output in each state, so its degree of exploration is low and its convergence result is not ideal. On the contrary, the TCD-SAC algorithm with a stochastic policy is indeed superior in UAV

penetration. When the TCD-PPO algorithm is used, its convergence speed is similar to that of the TCD-SAC algorithm, but its convergence result is not as good as that of the TCD-SAC algorithm. This is because on-policy algorithms such as the TCD-PPO algorithm cannot be used while exploring, so it is easy to fall into local optimization. On the contrary, off-policy algorithms such as the TCD-SAC algorithm are superior in UAV penetration. Among the above three algorithms combined with the TCD method, the TCD-SAC algorithm performs the best, which proves the superiority of the SAC algorithm in UAV penetration.

In addition, the success rate of penetrations for each algorithm is shown in Figure 9, and the success rate of hits for each algorithm is shown in Figure 10. Their change with episodes is consistent with the average reward in Figure 8, which shows the rationality and correctness of reward shaping.

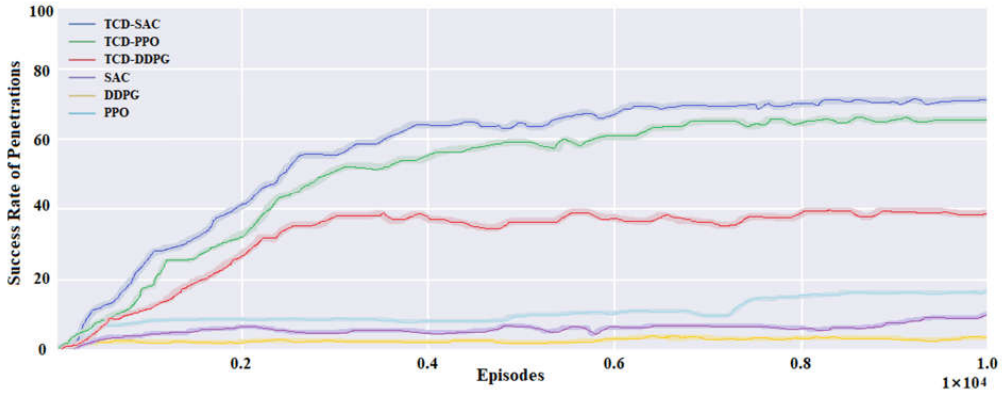


Figure 9. The success rate of penetrations for each algorithm.

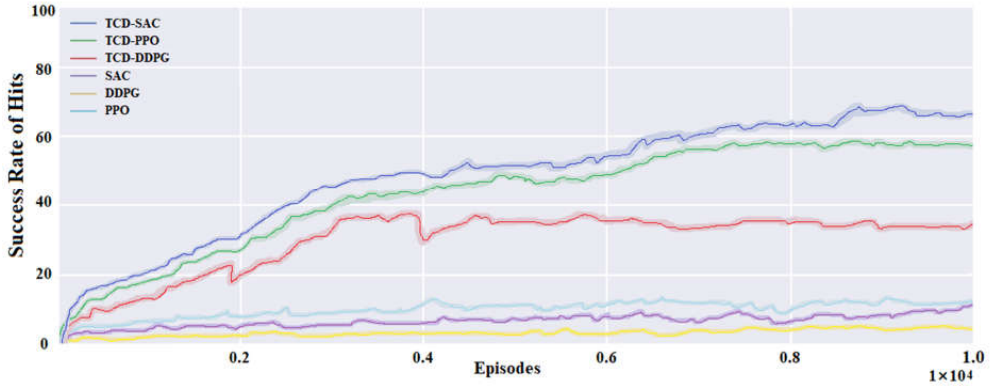
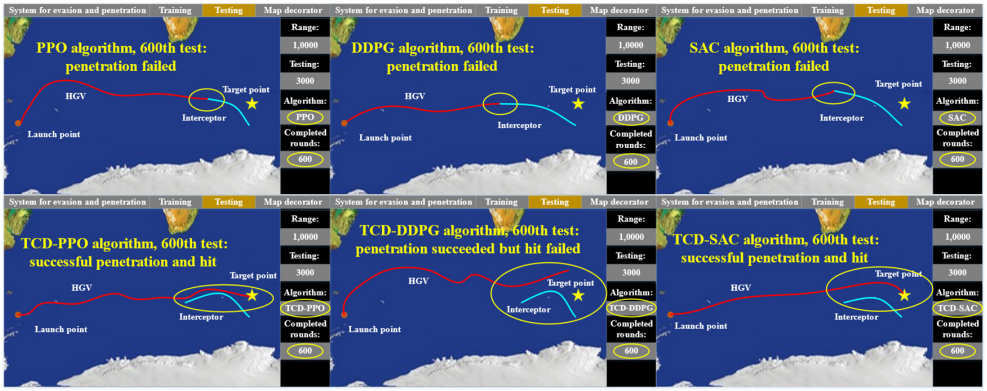


Figure 10. The success rate of hits for each algorithm.

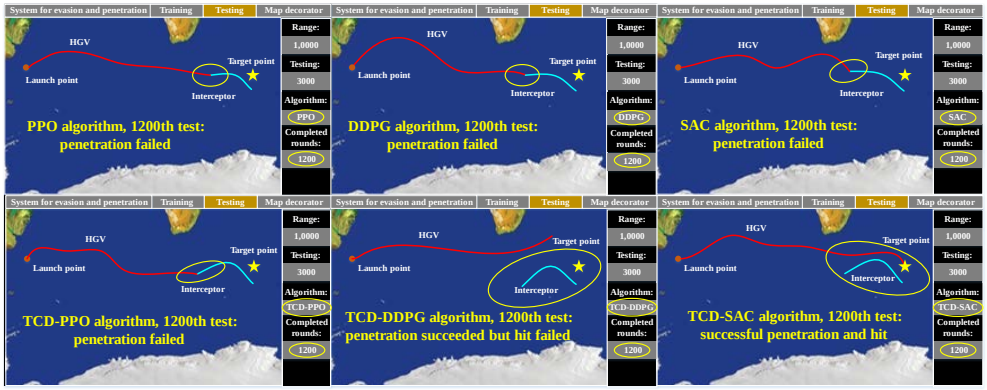
The above is the result of training experiments. To test the actual performance of each algorithm further, a test experiment should later be conducted, which is shown in Section 5.3.

5.3. Performance in Test Experiment

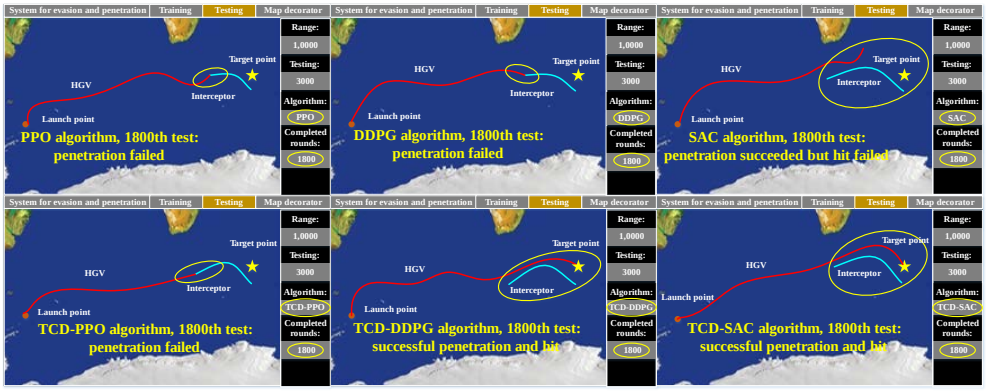
The maximum number of test experiments is 3000, and the launch point and target point are randomly initialized in each test to verify the general performance of all algorithms in a complex dynamic environment. Since it is impossible to show all test results here, the 600th, 1200th, 1800th, 2400th, and 3000th test results are selected, respectively, corresponding to Figure a–e in Figure 11. In Figure 11, the red line is the trajectory of the UAV, and the blue line is the trajectory of the interceptor. The performance of the UAV with different algorithms in the test environment is shown in Figure 11, which is from a two-dimensional perspective.



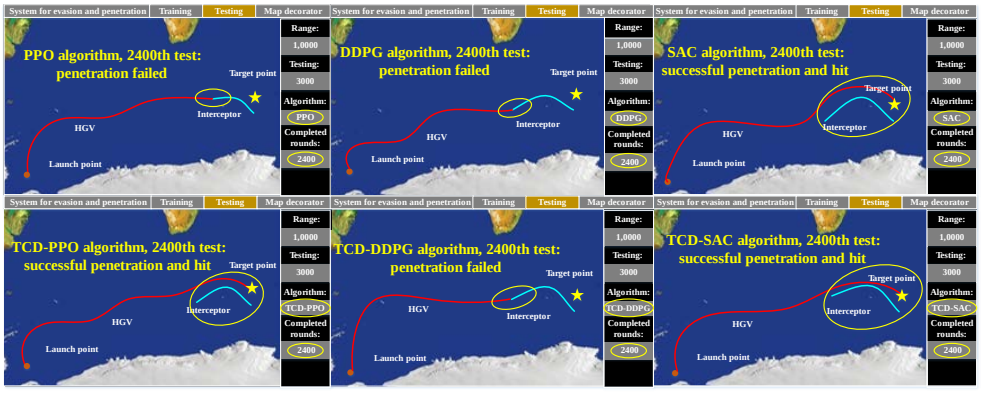
(**a**)



(**b**)

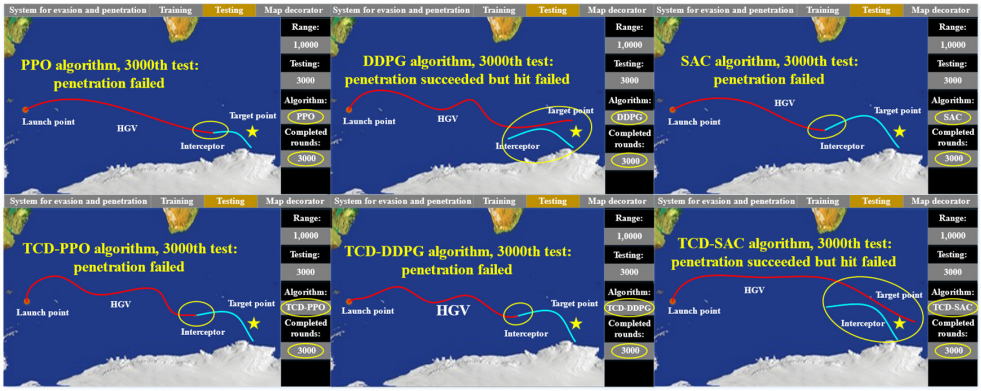


(**c**)



(**d**)

Figure 11. *Cont.*



(**e**)

Figure 11. The performance of different algorithms in the test environment. Among them, subfigure (*a*) represents the results of the 600th test, subfigure (**b**) represents the results of the 1200th test, subfigure (**c**) represents the results of the 1800th test, subfigure (**d**) represents the results of the 2400th test, and subfigure (**e**) represents the results of the 3000th test.

Figure 11a shows the result of the 600th test experiment. When the PPO algorithm, DDPG algorithm, or SAC algorithm is used, the test result is "penetration failed"; when the TCD-DDPG algorithm is used, the test result is "penetration succeeded but hit failed"; when the TCD-PPO algorithm or TCD-SAC algorithm is used, the test result is "successful penetration and hit".

Figure 11b shows the result of the 1200th test experiment. When the PPO algorithm, DDPG algorithm, SAC algorithm, or TCD-PPO algorithm is used, the test result is "penetration failed"; when the TCD-DDPG algorithm is used, the test result is "penetration succeeded but hit failed"; when the TCD-SAC algorithm is used, the test result is "successful penetration and hit".

Figure 11c shows the result of the 1800th test experiment. When the PPO algorithm, DDPG algorithm, or TCD-PPO algorithm is used, the test result is "penetration failed"; when the SAC algorithm is used, the test result is "penetration succeeded but hit failed"; when the TCD-DDPG algorithm or TCD-SAC algorithm is used, the test result is "successful penetration and hit".

Figure 11d shows the result of the 2400th test experiment. When the PPO algorithm, DDPG algorithm, or TCD-DDPG algorithm is used, the test result is "penetration failed"; when the SAC algorithm, TCD-PPO algorithm, or TCD-SAC algorithm is used, the test result is "successful penetration and hit".

Figure 11e shows the result of the 3000th test experiment. When the PPO algorithm, SAC algorithm, TCD-PPO algorithm, or TCD-DDPG algorithm is used, the test result is "UAV penetration failed"; when the DDPG algorithm or TCD-SAC algorithm is used, the test result is "penetration succeeded but hit failed"; when the TCD-DDPG algorithm or TCD-SAC algorithm is used, the test result is "successful penetration and hit".

In the five test results above, the TCD-SAC algorithm has four perfect results of "successful penetration and hit" and only one bad result of "penetration succeeded but hit failed", which is because the landing point of the UAV slightly deviated from the target point. On the contrary, other algorithms mostly have bad results of "penetration failed" or "penetration succeeded but hit failed". Therefore, the TCD-SAC algorithm proposed in this paper has the best performance in the above test results.

The five results in Figure 11 are only a sample of 3000 test results. To objectively evaluate the performance of each algorithm, the 3000 test results should be comprehensively evaluated. Therefore, this paper draws the minimum distance between the UAV and interceptor in each of the 3000 tests, which is the index to evaluate whether the UAV has successfully penetrated. As mentioned earlier, once the minimum distance between the

UAV and interceptor is less than 300 m, it is considered a failed penetration; otherwise, it is considered a successful penetration.

Figure 12 illustrates that when the PPO, SAC, or DDPG algorithm is tested, the minimum distance between the UAV and interceptor (hereinafter referred to as mini-distance) is less than 100 m in each test, which means failed penetration. When the TCD-DDPG algorithm is tested, the mini-distance increases to about 250 m, but its mini-distance is still less than 300 m, so most results are still failed penetration; when the TCD-PPO algorithm is tested, its mini-distance increases to about 280 m, which is still less than 300 m and most results are failed penetration; when the TCD-SAC algorithm is tested, the mini-distance increases to about 330 m. In this case, the mini-distance of most tests is greater than 300 m, so most results are successful penetration. Therefore, the TCD-SAC algorithm proposed in this paper has the best performance of penetration in 3000 tests.

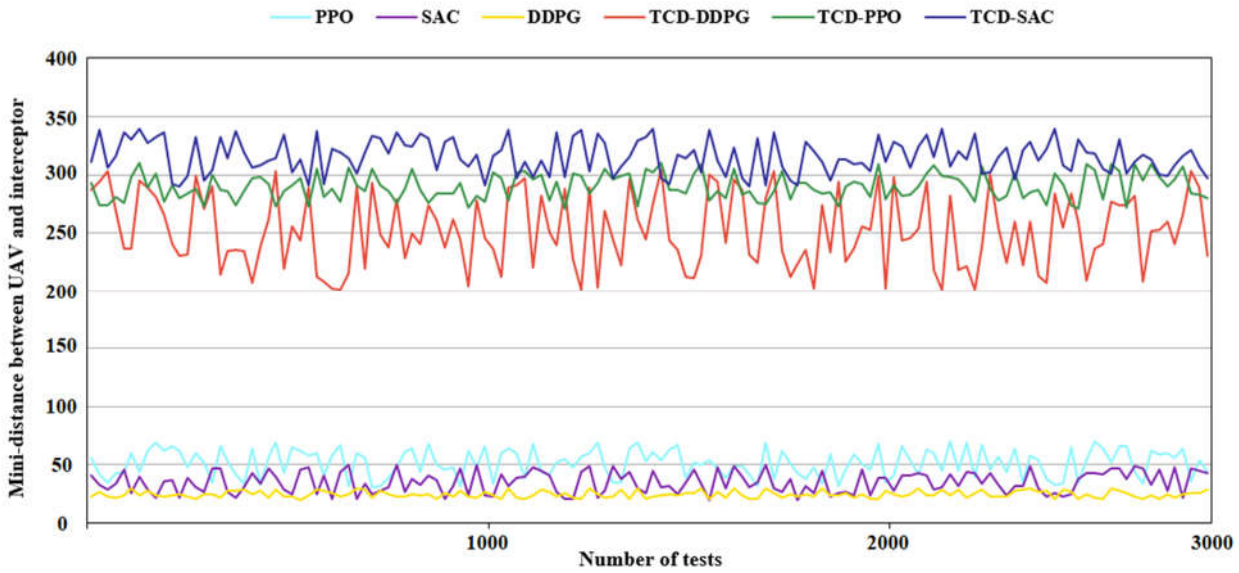


Figure 12. The minimum distance between the UAV and interceptor in 3000 tests.

For the convenience of description, this paper defines the number of tests as NT, the number of successful penetrations as NP, the number of successful hits as NH, the success rate of penetrations as RP, and the success rate of hits as RH. RP is equal to NP divided by NT, and RH is equal to NH divided by NT. Based on the results in Figure 12, the success rate of penetrations (RP) in 3000 tests is shown in Figure 13, and the success rate of hits (RH) in 3000 tests is shown in Figure 14. From Figures 13 and 14, the success rates of penetrations (RP) of DDPG, PPO, and SAC algorithms are 0.63%, 3.27%, and 3.50%, respectively, which are all less than 4%, and their success rates of hits (RH) are 0.47%, 2.80%, and 2.87%, respectively, which are all less than 3%. The above results show that these algorithms have a poor performance in UAV penetration. In contrast, the RP and RH of the TCD-DDPG algorithm are 17.83% and 11.13%, respectively, and those of the TCD-PPO algorithm are 42.90% and 31.97%, respectively. Compared to the first three algorithms, both the TCD-DDPG and TCD-PPO algorithms exhibit significant performance improvements. Furthermore, the RP and RH of the TCD-SAC algorithm are 68.20% and 67.90%, respectively, whose performance is far ahead of the other algorithms. That is to say, when the TCD-SAC algorithm is tested, all subtasks are completed perfectly in most of the 3000 tests, indicating the reliability and universality of the algorithm.

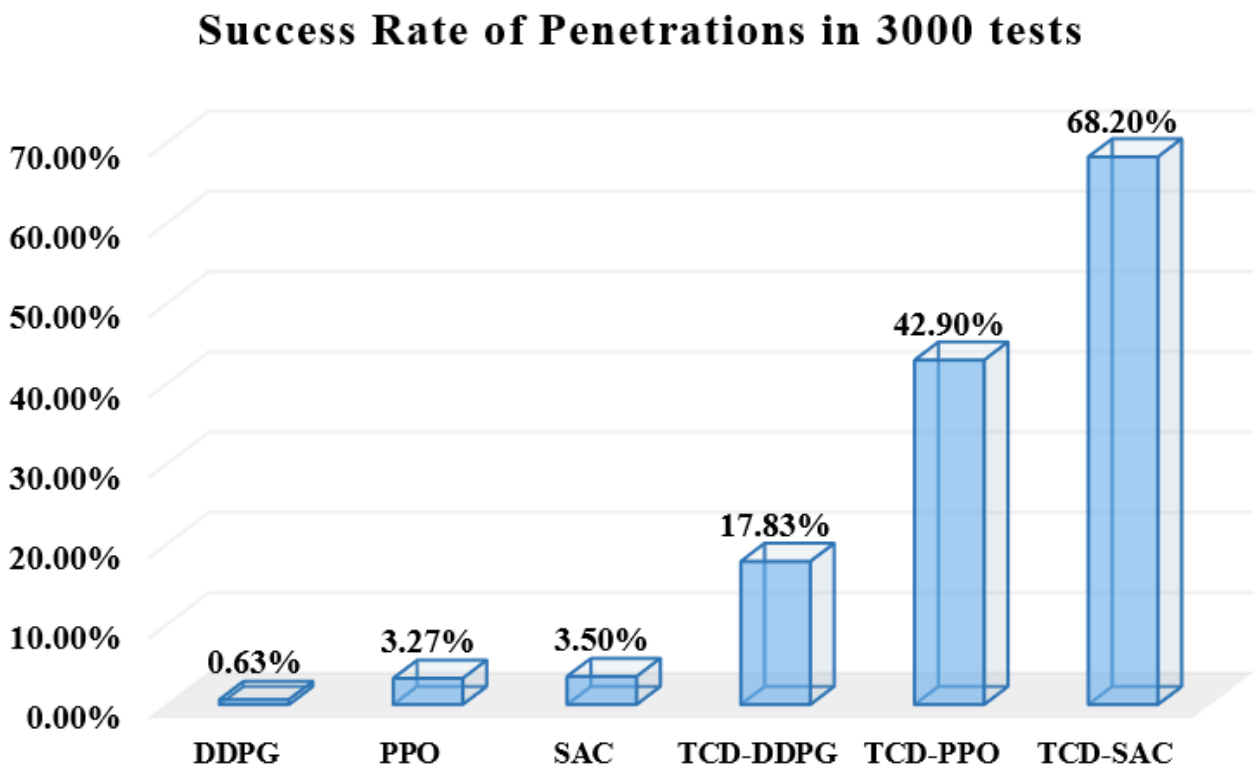


Figure 13. The success rate of penetrations in 3000 tests.

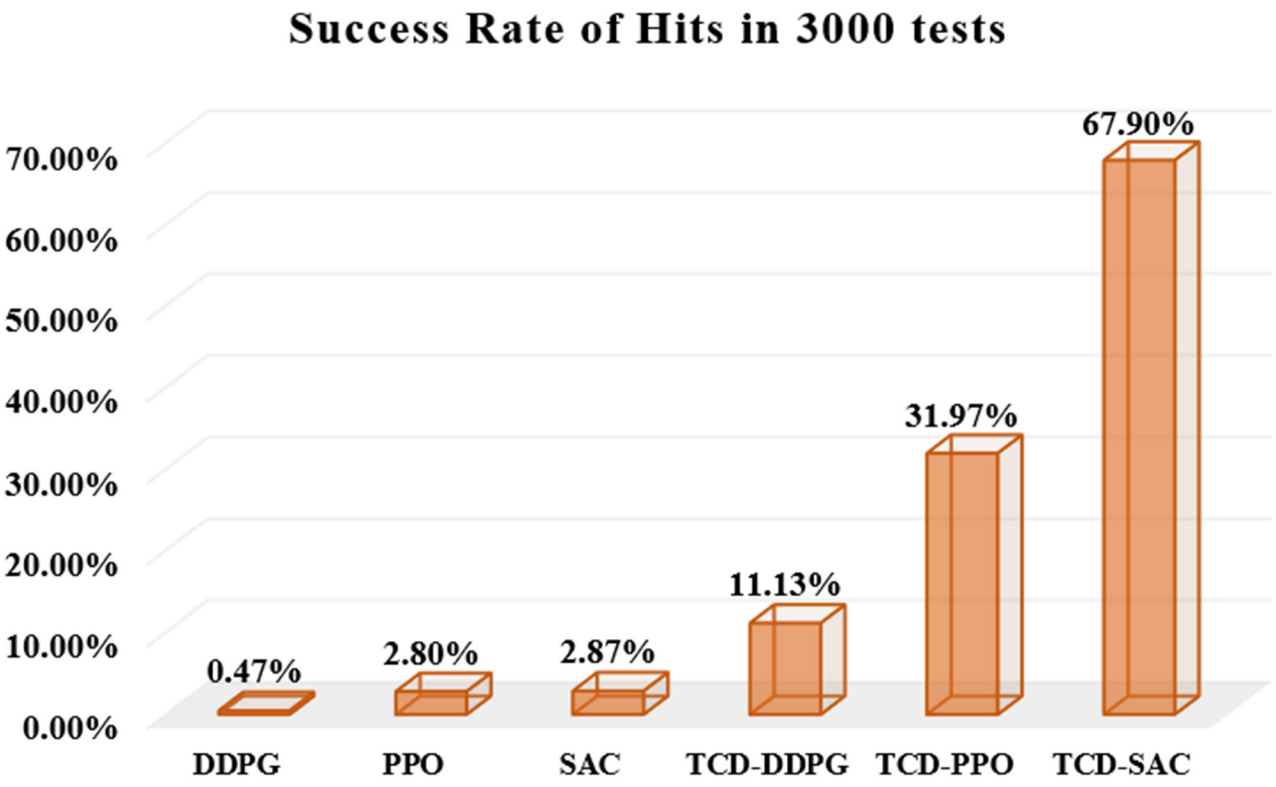


Figure 14. The success rate of hits in 3000 tests.

The above test results show that the TCD-SAC algorithm obtained from training has a good feasibility and robustness in a complex dynamic environment. When the UAV encounters the interceptor, the penetration policy output by the TCD-SAC algorithm can play a significant role in time, quickly guiding the UAV to maneuver and avoid the interceptor to complete the task of autonomous penetration and hitting the target.

6. Conclusions and Future Work

Committed to improving the sample utilization of UAV penetration, this paper proposes the improved sampling method TCD and combines this TCD method with the SAC algorithm to obtain the TCD-SAC algorithm. To verify the real performance of the TCD-SAC algorithm, this study designs corresponding action space, state space, and a reward function, and conducts training experiments and test experiments for UAV penetration. All experiments are carried out in a complex dynamic environment. Compared with the original SAC algorithm and two other excellent algorithms (DDPG and PPO), the training experiments show that the improved TCD-SAC algorithm has the fastest convergence speed and the best convergence result in all algorithms. The test experiments show that the improved TCD-SAC algorithm has the highest success rate of penetrations and hits, illustrating the feasibility and robustness of the TCD-SAC algorithm. This paper presents a relatively complete study from model building to algorithm improvement to experimental validation, which provides potentially useful ideas for the future autonomous penetration of UAVs.

Of course, because the cost of training and testing a real physical UAV is too high, this paper only conducted experiments in a simulation environment. If the algorithm is to be applied to a real physical UAV, it will be necessary to eliminate the difference between the real model and simulation model as much as possible. Therefore, our next task is to try to bridge the reality gap between a real model and simulation model. For example, when an algorithm performs well in a simulation environment, methods such as transfer learning can be used to transfer the trained algorithm to a real environment quickly and cost effectively.

Author Contributions: Conceptualization, Y.W. and X.Z.; methodology, X.Z. and K.L.; software, X.L. and H.L.; validation, K.L.; formal analysis, K.L.; investigation, K.L.; resources, K.L. and Y.W.; data curation, K.L.; writing—original draft preparation, X.Z. and Y.W.; writing—review and editing, K.L., X.L., and H.L.; visualization, K.L.; supervision, X.L. and H.L.; project administration, Y.W.; funding acquisition, K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

Appendix A

The SAC algorithm defines the policy in the Actor Network as $\pi_\phi(a_t|s_t)$ and defines the output of the Q Critic Network as $Q_\theta(s_t, a_t)$. The update method of the Q Critic Network is shown in Formula (A1).

$$\begin{aligned} J_Q(\theta) &= \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma V_{\bar{\theta}}(s_{t+1})\right)\right)^2\right] \\ &= \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D},a_t\sim\pi_\phi}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma(Q_{\bar{\theta}}(s_{t+1},a_{t+1}) - \alpha\log\left(\pi_\phi(s_{t+1},a_{t+1})\right)\right)\right)\right)^2\right] \end{aligned} \tag{A1}$$

where $J_Q(\theta)$ is the cost function of the Q Critic Network, s_t is the current state, a_t is the current action, $\mathcal{D}$ is the replay pool, s_{t+1} is the next state, $r(s_t, a_t)$ is the corresponding reward, γ is the discount factor, θ and ϕ are parameter vectors of the Q Critic Network, and $\bar{\theta}$ is obtained from θ through an exponentially moving average.

The update method of the Actor Network is to minimize the Kullback–Leibler Divergence, which is shown in Formula (A2).

$$\begin{aligned} J_\pi(\phi) &= D_{KL}(\pi_\phi(.|s_t)||\exp\left(\tfrac{1}{\alpha}Q_\theta(s_t,.) - logZ(s_t)\right)) \\ &= \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\phi}[\log(\tfrac{\pi_\phi(a_t|s_t)}{\exp(\frac{1}{\alpha}Q_\theta(s_t,a_t)-logZ(s_t))})] \\ &= \mathbb{E}_{s_t\sim\mathcal{D},a_t\sim\pi_\phi}[log\pi_\phi(a_t|s_t) - \tfrac{1}{\alpha}Q_\theta(s_t,a_t) + logZ(s_t)] \end{aligned} \tag{A2}$$

where $J_\pi(\phi)$ is the cost function of the Actor Network, and the generation method of a_t is shown in Formula (A3).

$$a_t = f_\phi(\varepsilon_t; s_t) = f_\phi^\mu(s_t) + \varepsilon_t \odot f_\phi^\sigma(s_t) \tag{A3}$$

where the f function calculates the mean and variance of the sample, the sample follows a standard normal distribution, and ε is the noise of the standard normal distribution. Based on the reparameterization trick in Formula (A3), the whole process of Formula (A2) is completely differentiable. Therefore, Formula (A2) can be simplified into Formula (A4).

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\mathcal{D},\varepsilon\sim\mathcal{N}}\left[\alpha log\pi_\phi\left(f_\phi(\varepsilon_t;s_t)|s_t\right) - Q_\theta\left(s_t, f_\phi(\varepsilon_t;s_t)\right)\right] \tag{A4}$$

References

1. Kumar, G.N.; Sarkar, A.; Mangrulkar, K.; Talole, S. Atmospheric vehicle trajectory optimization with minimum dynamic pressure constraint. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2018**, *232*, 837–846. [CrossRef]
2. Chai, R.; Savvaris, A.; Tsourdos, A.; Chai, S.; Xia, Y.; Wang, S. Solving trajectory optimization problems in the presence of probabilistic constraints. *IEEE Trans. Cybern.* **2019**, *50*, 4332–4345. [CrossRef]
3. Xu, J.; Dong, C.; Cheng, L. Deep neural network-based footprint prediction and attack intention inference of hypersonic glide vehicles. *Mathematics* **2022**, *11*, 185. [CrossRef]
4. Gao, Y.; Cai, G.; Yang, X.; Hou, M. Improved tentacle-based guidance for reentry gliding hypersonic vehicle with no-fly zone constraint. *IEEE Access* **2019**, *7*, 119246-58. [CrossRef]
5. Liao, Y.; Li, H. Trajectory optimization for terminal phase flight of hypersonic reentry vehicles with multi-constraints. In Proceedings of the 2013 25th Chinese Control and Decision Conference (CCDC), Guiyang, China, 25–27 May 2013; pp. 571–576.
6. Sana, K.S.; Weiduo, H. Hypersonic reentry trajectory planning by using hybrid fractional-order particle swarm optimization and gravitational search algorithm. *Chin. J. Aeronaut.* **2021**, *34*, 50–67. [CrossRef]
7. Chai, R.; Savvaris, A.; Tsourdos, A.; Chai, S.; Xia, Y. Improved gradient-based algorithm for solving aeroassisted vehicle trajectory optimization problems. *J. Guid. Control Dyn.* **2017**, *40*, 2093–2101. [CrossRef]
8. Wan, K.; Gao, X.; Hu, Z.; Wu, G. Robust motion control for uav in dynamic uncertain environments using deep reinforcement learning. *Remote Sens.* **2020**, *12*, 640. [CrossRef]
9. Luo, J.; Liang, Q.; Li, H. Uav penetration mission path planning based on improved holonic particle swarm optimization. *J. Syst. Eng. Electron.* **2023**, *34*, 197–213. [CrossRef]
10. Fu, J.; Sun, G.; Yao, W.; Wu, L. On trajectory homotopy to explore and penetrate dynamically of multi-uav. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 24008–24019. [CrossRef]
11. Zhang, Z.; Wu, J.; Dai, J.; He, C. A novel real-time penetration path planning algorithm for stealth uav in 3d complex dynamic environment. *IEEE Access* **2020**, *8*, 122757–122771. [CrossRef]
12. Luo, Y.; Song, J.; Zhao, K.; Liu, Y. Uav-cooperative penetration dynamic-tracking interceptor method based on ddpg. *Appl. Sci.* **2022**, *12*, 1618. [CrossRef]
13. Li, Y.; Han, W.; Wang, Y. Deep reinforcement learning with application to air confrontation intelligent decision-making of manned/unmanned aerial vehicle cooperative system. *IEEE Access* **2020**, *8*, 67887–67898. [CrossRef]
14. Kaifang, W.; Bo, L.; Xiaoguang, G.; Zijian, H.; Zhipeng, Y. A learning-based flexible autonomous motion control method for uav in dynamic unknown environments. *J. Syst. Eng. Electron.* **2021**, *32*, 1490–1508. [CrossRef]
15. Liang, L.; Deng, F.; Wang, J.; Lu, M.; Chen, J. A reconnaissance penetration game with territorial-constrained defender. *IEEE Trans. Autom. Control* **2022**, *67*, 6295–6302. [CrossRef]
16. Bellman, R. A markovian decision process. *J. Math. Mech.* **1957**, *6*, 679–684. [CrossRef]
17. Li, S.; Lei, H.; Shao, L.; Xiao, C. Multiple model tracking for hypersonic gliding vehicles with aerodynamic modeling and analysis. *IEEE Access* **2019**, *7*, 28011–28018. [CrossRef]
18. Liu, X.; Zhang, Y.; Wang, S.; Huang, W. Backstepping attitude control for hypersonic gliding vehicle based on a robust dynamic inversion approach. *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.* **2014**, *228*, 543–552. [CrossRef]
19. Li, G.; Zhang, H.; Tang, G. Maneuver characteristics analysis for hypersonic glide vehicles. *Aerosp. Sci. Technol.* **2015**, *43*, 321–328. [CrossRef]

20. Shen, Z.; Yu, J.; Dong, X.; Hua, Y.; Ren, Z. Penetration trajectory optimization for the hypersonic gliding vehicle encountering two interceptors. *Aerosp. Sci. Technol.* **2022**, *121*, 107363. [CrossRef]
21. Yan, B.; Liu, R.; Dai, P.; Xing, M.; Liu, S. A rapid penetration trajectory optimization method for hypersonic vehicles. *International J. Aerosp. Eng.* **2019**, *2019*, 11. [CrossRef]
22. Chai, R.; Tsourdos, A.; Savvaris, A.; Chai, S.; Xia, Y.; Chen, C.P. Six-dof spacecraft optimal trajectory planning and real-time attitude control: A deep neural network-based approach. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 5005–5013. [CrossRef] [PubMed]
23. Xiang, J.; Li, Q.; Dong, X.; Ren, Z. Continuous control with deep reinforcement learning for mobile robot navigation. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; pp. 1501–1506.
24. Wenjun, N.; Ying, B.; Di, W.; Xiaoping, M. Energy-optimal trajectory planning for solar-powered aircraft using soft actor-critic. *Chin. J. Aeronaut.* **2022**, *35*, 337–353.
25. Eteke, C.; Kebüde, D.; Akgün, B. Reward learning from very few demonstrations. *IEEE Trans. Robot.* **2020**, *37*, 893–904. [CrossRef]
26. Han, S.-C.; Bang, H.; Yoo, C.-S. Proportional navigation-based collision avoidance for uavs. *Int. J. Control Autom. Syst.* **2009**, *7*, 553. [CrossRef]
27. Wang, Z.; Cheng, X.X.; Li, H. Hypersonic skipping trajectory planning for high l/d gliding vehicles. In Proceedings of the 21st AIAA International Space Planes and Hypersonics Technologies Conference, Xiamen, China, 6–9 March 2017; p. 2135.
28. Tripathi, A.K.; Patel, V.V.; Padhi, R. Autonomous landing of fixed wing unmanned aerial vehicle with reactive collision avoidance. *IFAC-Pap.* **2018**, *51*, 474–479. [CrossRef]
29. Maeda, R.; Mimura, M. Automating post-exploitation with deep reinforcement learning. *Comput. Secur.* **2021**, *100*, 102108. [CrossRef]
30. Sackmann, M.; Bey, H.; Hofmann, U.; Thielecke, J. Modeling driver behavior using adversarial inverse reinforcement learning. In Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany, 5–9 June 2022; pp. 1683–1690.
31. Hu, Y.; Gao, C.; Li, J.; Jing, W.; Li, Z. Novel trajectory prediction algorithms for hypersonic gliding vehicles based on maneuver mode on-line identification and intent inference. *Meas. Sci. Technol.* **2021**, *32*, 115012. [CrossRef]
32. Alzahrani, B.; Oubbati, O.S.; Barnawi, A.; Atiquzzaman, M.; Alghazzawi, D. Uav assistance paradigm: State-of-the-art in applications and challenges. *J. Netw. Comput. Appl.* **2020**, *166*, 102706. [CrossRef]
33. Kontogiannis, S.G.; Ekaterinaris, J.A. Design, performance evaluation and optimization of a uav. *Aerosp. Sci. Technol.* **2013**, *29*, 339–350. [CrossRef]
34. Zhang, S.; Zhang, H.; Di, B.; Song, L. Cellular uav-to-x communications: Design and optimization for multi-uav networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 1346–1359. [CrossRef]
35. Oubbati, O.S.; Atiquzzaman, M.; Ahanger, T.A.; Ibrahim, A. Softwarization of uav networks: A survey of applications and future trends. *IEEE Access* **2020**, *8*, 98073–98125. [CrossRef]
36. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement learning for uav attitude control. *ACM Trans. Cyber-Phys. Syst.* **2019**, *3*, 1–21. [CrossRef]