Journal of Intelligent & Robotic Systems (2022) 104: 60
https://doi.org/10.1007/s10846-022-01601-8

REGULAR PAPER

Autonomous Obstacle Avoidance and Target Tracking of UAV Based on Deep Reinforcement Learning

Guoqiang Xu[1] · Weilai Jiang[1] · Zhaolei Wang[2] · Yaonan Wang[1]

Received: 4 October 2021 / Accepted: 21 February 2022 / Published online: 19 March 2022

Abstract
When using deep reinforcement learning algorithm to complete Unmanned Aerial Vehicle (UAV) autonomous obstacle avoidance and target tracking tasks, there are often some problems such as slow convergence speed and low success rate. Therefore, this paper proposes a new deep reinforcement learning algorithm, namely Multiple Pools Twin Delay Deep Deterministic Policy Gradient (MPTD3) algorithm. Firstly, the state space and action space of UAV are established as continuous models, which is closer to engineering practice than discrete models. Then, multiple experience pools mechanism and gradient truncation are designed to improve the convergence of the algorithm. Furthermore, the generalization ability of the algorithm is obtained by giving UAV environmental perception ability. Experimental results verify the effectiveness of the proposed method.

Keywords Unmanned Aerial Vehicle (UAV) · Autonomous obstacle avoidance · Target tracking · Deep reinforcement learning · Continuous control

1 Introduction

Unmanned Aerial Vehicle (UAV) has the characteristics of small volume, light weight, low cost and strong mobility, so it has been widely used in military and civil fields, such as logistics transportation [1–4], disaster rescue [5–8], relay communication [9–12] and power inspection [13, 14]. Among them, autonomous obstacle avoidance and target tracking of UAV are the primary problems to be focused in engineering application, which have been deeply studied by many scholars at home and abroad.

Autonomous obstacle avoidance and target tracking of UAV in complex environment is a typical decision-making problem. Reinforcement learning has obvious advantages in solving such problems, so it has attracted extensive attention in this field. Reinforcement learning is one of the three paradigms of machine learning. It uses the empirical data generated by the continuous interaction between the agent and the environment to update the agent's policy, so as to maximize the cumulative reward. In [15], Q-learning algorithm is used for static target tracking of UAV in discrete environment. However, Q-learning algorithm is only suitable for solving discrete low dimensional state space problems, without considering the continuous change of states, so it has some limitations in engineering.

Deep reinforcement learning is the combination of deep learning and reinforcement learning. It not only has the strong representation ability of deep learning, but also has the excellent decision-making ability of reinforcement learning. Deep Q-Network (DQN) [16] is the first deep reinforcement learning algorithm. It was improved and proposed by DeepMind team on the basis of Q-learning algorithm, and has reached the level of human players or even surpassed human players in many video games. Dueling Double Deep Q-Network (D3QN) algorithm is a further improvement of DQN algorithm. In [17], D3QN algorithm and convolutional neural network are combined to solve the problems of autonomous obstacle avoidance and target tracking of UAV in two-dimensional environment. The artificial potential field method is an effective path planning method. In [18], the black hole

✉ Weilai Jiang
jiangweilai@hnu.edu.cn

Guoqiang Xu
xuguoqiang@hnu.edu.cn

[1] College of Electrical and Information Engineering, Hunan University, Changsha 410082, China

[2] Science and Technology on Aerospace Intelligent Control Laboratory, Beijing Aerospace Automatic Control Institute, Beijing 100854, China

artificial potential field method is designed to construct the reward function in the DQN algorithm to solve the local optimization problem and realize the autonomous obstacle avoidance and target search of UAV in complex environment. Deep Recurrent Q-Network (DRQN) [19] is obtained by adding Long Short Term Memory (LSTM) to the network of DQN algorithm, which is suitable for solving partially observable problems. In [20], it is used to solve the autonomous obstacle avoidance problem of UAV in a limited environment. Although the above algorithm can be used to solve the continuous state space, it cannot deal with the continuous action space, so it still has some limitations. Deep Deterministic Policy Gradient (DDPG) algorithm [21] is an important algorithm under the framework of actor-critic, which can effectively solve the problems of continuous state space and action space. In [22–24], DDPG algorithm is adopted for UAV decision-making. Its input is original state of the surrounding environment and output is deterministic action. In [25], the target recognition and detection network and DDPG algorithm are combined to make decision by using the position information of surrounding obstacles, so as to realize the search and rescue of UAV in complex indoor environment. In the complex environment, some circular obstacles are difficult to cross, and UAVs are easy to fall into local optimal solutions. In [26], the Recurrent Deterministic Policy Gradient (RDPG) algorithm [27] is employed, which takes the past historical state information and action information as network input and significantly improves the navigation and obstacle avoidance ability of UAV. Twin Delay Deep Deterministic Policy Gradient (TD3) algorithm [28] is an improvement of DDPG algorithm, which effectively avoids the over estimation problem under the framework of actor-critic and improves the overall performance of the algorithm. In [29], the TD3 algorithm and meta learning are combined to realize dynamic target tracking of UAV in obstacle free environment. The above literatures mainly discuss the autonomous obstacle avoidance and target tracking of UAV in static environment or autonomous obstacle avoidance in dynamic environment, but there are few reports on the autonomous obstacle avoidance and dynamic target tracking of UAV in complex environment.

This paper mainly studies the autonomous obstacle avoidance and dynamic target tracking of UAV in complex environment. Its contributions are reflected in three aspects.

1. Different from the discretization of state space or action space in [15–20], this paper considers that both state and action of UAV change continuously, which is closer to engineering practice.
2. Based on the TD3 algorithm, a multiply experience pools mechanism is designed to divide the experience data generated by the interaction between UAV and environment into successful experience data and failed experience data, so as to improve the convergence performance and tracking success rate of the algorithm.
3. The UAV is endowed with environmental perception ability, and the overall observation information is obtained by integrating environmental perception information and target state information, so as to improve the generalization ability of the algorithm.

The next arrangement of the paper is as follows: Section 2 describes the autonomous obstacle avoidance and target tracking of UAV in detail. Section 3 establishes the continuous dynamic model of UAV. Section 4 introduces TD3 algorithm and proposes multiple experience pools mechanism. In Section 5, the proposed algorithm is verified and analyzed. Section 6 is the summary of the full paper and points out the shortcomings.

2 Problem Description

In order to facilitate the analysis and solution of the problem, this paper considers that the UAV is always flying at a fixed height and regarded as a particle, so the motion mode of the UAV can be simplified from three-dimensional space motion to two-dimensional plane translation. In addition, the shapes of obstacles are regarded as circles with different radius in the environment, and the UAV is set to move in a rectangular area. The schematic diagram of UAV autonomous obstacle avoidance and target tracking is shown in Fig. 1.

In Fig. 1, the blue UAV represents the pursuer, namely the agent, which tracks the target in real time and automatically avoids obstacles in the environment. The red UAV indicates

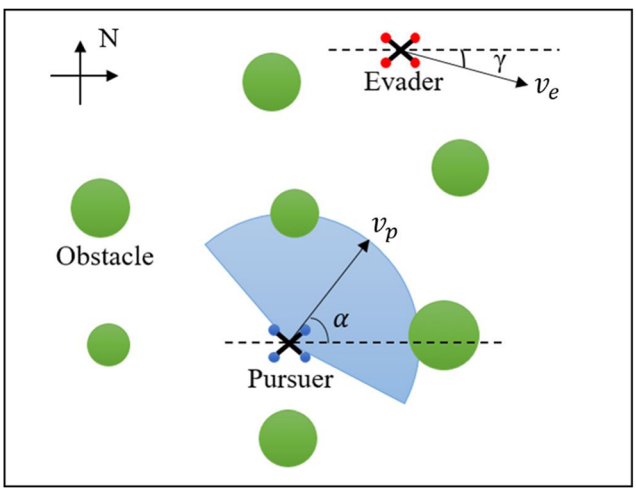


Fig. 1 Schematic diagram of UAV autonomous obstacle avoidance and target tracking. The blue UAV represents the pursuer, the red UAV represents the evader, the green circle represents the obstacle, and the cyan area represents the pursuer's perception range

the evader, namely the tracking target, which may be in a static or random motion state in the environment and can also avoid obstacles automatically. Green circles indicate obstacles, the number and location of them in the environment are uncertain. When the UAV moves in a two-dimensional environment, its own state information includes position coordinates, speed magnitude and motion angle. The control quantity is the change of speed and angle. In order to ensure the stable operation of UAV, its control quantity is within a certain threshold. In order to improve the autonomous obstacle avoidance ability of the UAV, it can sense the surrounding environment by relying on its equipped camera or laser radar. The cyan area in Fig. 1 is the perception range of the UAV. Through environmental perception, UAV can obtain partial environmental information around itself. In addition, the pursuer can also obtain the state information of the evader through Global Positioning System (GPS) or other devices.

The research goal of this paper is to enable the pursuer can make real-time decisions by using collected information, so as to realize its autonomous obstacle avoidance and target tracking in different complexity environments after training in a simple environment.

3 Dynamic Model

Based on the deep reinforcement learning method, this paper establishes the model of UAV autonomous obstacle avoidance and target tracking. Since UAV can only perceive part of the environmental information and cannot observe the global state information in the environment, the pursuer is modeled as a Partially Observable Markov Decision Process (POMDP) in this paper. The state space, action space and reward function in the model are defined below.

3.1 Observation Space

The state information of UAV includes two-dimensional coordinates, speed magnitude and motion angle. It is assumed that the position coordinate of the pursuer at time t is $\left(x_{\mathrm{p}}^t, y_{\mathrm{p}}^t\right)$, the speed magnitude is v_{p}^t, and the motion angle is θ_{p}^t. The position coordinate of the evader is $\left(x_{\mathrm{e}}^t, y_{\mathrm{e}}^t\right)$, the speed magnitude is v_{e}^t, and the motion angle is θ_{e}^t, as shown in Fig. 2. Considering the flight speed limit of UAV, we set the maximum flight speed of pursuer as $v_{\mathrm{p}}^{\max}$ and the maximum flight speed of evader as $v_{\mathrm{e}}^{\max}$.

In order to eliminate the influence of different dimensions, the state information of pursuer and evader are normalized respectively, as shown in Eq. 1.

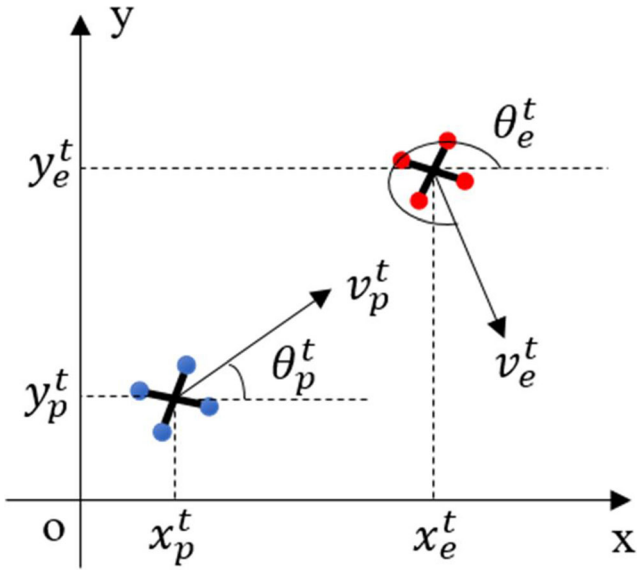


Fig. 2 Schematic diagram of UAV state. The state information of UAV includes position, speed and angle

$$
\begin{cases}
o_1^t = \dfrac{x_{\mathrm{p}}^t}{w}, x_{\mathrm{p}}^t \in [0, w] \\
o_2^t = \dfrac{y_{\mathrm{p}}^t}{h}, y_{\mathrm{p}}^t \in [0, h] \\
o_3^t = \dfrac{\theta_{\mathrm{p}}^t \bmod (2\pi)}{\pi} - 1, \theta_{\mathrm{p}}^t \in [0, 2\pi] \\
o_4^t = \dfrac{v_{\mathrm{p}}^t}{v_{\mathrm{p}}^{\max}}, v_{\mathrm{p}}^t \in \left[0, v_{\mathrm{p}}^{\max}\right] \\
o_5^t = \dfrac{x_{\mathrm{e}}^t}{w}, x_{\mathrm{e}}^t \in [0, w] \\
o_6^t = \dfrac{y_{\mathrm{e}}^t}{h}, y_{\mathrm{e}}^t \in [0, h] \\
o_7^t = \dfrac{\theta_{\mathrm{e}}^t \bmod (2\pi)}{\pi} - 1, \theta_{\mathrm{e}}^t \in [0, 2\pi] \\
o_8^t = \dfrac{v_{\mathrm{e}}^t}{v_{\mathrm{e}}^{\max}}, v_{\mathrm{e}}^t \in \left[0, v_{\mathrm{e}}^{\max}\right]
\end{cases}
\tag{1}
$$

where, o_1^t, o_2^t, o_3^t and o_4^t are the normalized state information of the pursuer. o_5^t, o_6^t, o_7^t and o_8^t are the normalized state information of the evader. h and w are the length and width of the UAV motion area respectively.

The pursuer can perceive part of the surrounding environment and obtain the distance information between himself and the obstacles. Within the sensing range, the UAV emits 9 rays around, and each ray returns the distance information obtained in the corresponding direction. When the ray does not detect any obstacles, returns the maximum measurement distance, as shown in Fig. 3.

In Fig. 3, d_1, d_2, $\cdots$, d_9 represent the distance of each ray direction, which meets

$$
d_i \in \left[0, r_{\mathrm{cap}}\right], (i = 1, 2, \cdots, 9) \tag{2}
$$

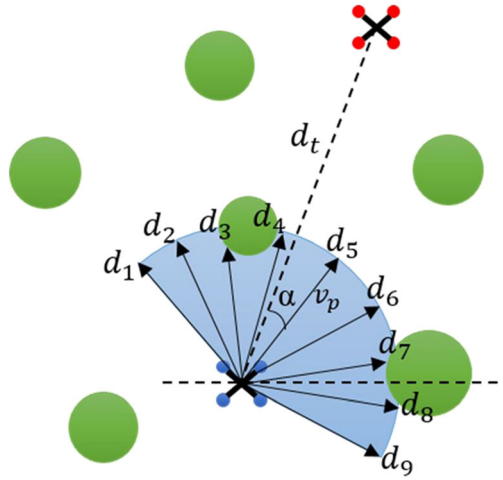


Fig. 3 Schematic diagram of UAV environment perception. The UAV sends out 9 rays, and each ray returns the distance information in corresponding direction

where, r_{cap} is the maximum distance that can be measured by the ray, that is, the maximum sensing range of the UAV.

α indicates the difference angle between the pursuer evader connection line and the pursuer motion direction, as shown in Eq. 3.

$$\begin{cases} \alpha = \arccos\left(\dfrac{\left(x_e^t - x_p^t\right)\cos\left(\theta_p^t\right) + \left(y_e^t - y_p^t\right)\sin\left(\theta_p^t\right)}{\sqrt{\left(x_e^t - x_p^t\right)^2 + \left(y_e^t - y_p^t\right)^2}}\right) \\ \alpha \in [-\pi, \pi] \end{cases} \tag{3}$$

d_t indicates the relative distance between the pursuer and the evader, as shown in Eq. 4. β indicates the relative angle between the pursuer-evader connection direction and the horizontal right direction.

$$\begin{cases} d_t = \sqrt{\left(x_e^t - x_p^t\right)^2 + \left(y_e^t - y_p^t\right)^2} \\ \beta = \arctan\left(\dfrac{y_e^t - y_p^t}{x_e^t - x_p^t}\right) \\ d_t \in \left[0, \sqrt{w^2 + h^2}\right] \\ \beta \in [-\pi, \pi] \end{cases} \tag{4}$$

Similarly, the above observation information is normalized to eliminate the dimensional influence, as shown in Eq. 5.

$$\begin{cases} o_j^t = \dfrac{d_i}{r_{cap}} (i = 1, 2, \cdots 9)(j = 9, 10, \cdots, 17) \\ o_{18}^t = \dfrac{\beta}{\pi} \\ o_{19}^t = \dfrac{d_t}{\sqrt{w^2 + h^2}} \\ o_{20}^t = \dfrac{\alpha}{\pi} \end{cases} \tag{5}$$

where, o_j^t is the normalized obstacle distance, o_{18}^t is the normalized relative angel, o_{19}^t is the normalized relative distance, and o_{20}^t is the normalized difference angle.

Therefore, the pursuer's observation space can be specifically expressed as

$$\mathbf{O} = \left[o_1^t, o_2^t, \cdots o_{19}^t, o_{20}^t\right] \tag{6}$$

3.2 Action Space

The motion control of the pursuer in two-dimensional space is divided into speed and angle. In order to ensure the smoothness of the pursuer's motion, the change of the control quantity is set within a certain range and continuously at each time step.

In this paper, the maximum change of the pursuer's speed at each time is Δv_{max}, and the maximum change of the angle is $\Delta\theta_{max}$. The action output by the pursuer at each time is a tuple, including the changes of speed and angle. Suppose that the action output by the pursuer at the current time is $[\lambda_\theta, \lambda_v](\lambda_\theta, \lambda_v \in [-1, 1])$, the updating method of its spatial position, speed and angle at the next time is

$$\begin{cases} v_p^{t+1} = v_p^t + \lambda_v \Delta v_{max} \\ \theta_p^{t+1} = \theta_p^t + \lambda_\theta \Delta\theta_{max} \\ x_p^{t+1} = x_p^t + v_p^{t+1}\cos\left(\theta_p^{t+1}\right) \\ y_p^{t+1} = y_p^t + v_p^{t+1}\sin\left(\theta_p^{t+1}\right) \end{cases} \tag{7}$$

where, θ_p^{t+1}, v_p^{t+1}, $\left(x_p^{t+1}, y_p^{t+1}\right)$ represent the angle, speed and position coordinates of the pursuer at the next time respectively.

Set the evader to do random movement in the environment, but there is still a certain movement mode. At the beginning, the birth position and movement speed of the evader are randomly initialized. The evader moves in a straight line along the speed direction in the environment. When encountering the environmental boundary, the speed change follows

$$\begin{cases} v_{ex}^{t+1} = \begin{cases} -v_{ex}^t, \text{left or right boundary} \\ v_{ex}^t, \quad \text{otherwise} \end{cases} \\ v_{ey}^{t+1} = \begin{cases} -v_{ey}^t, \text{upper or lower boundary} \\ v_{ey}^t, \quad \text{otherwise} \end{cases} \end{cases} \tag{8}$$

where, v_{ex}^t and v_{ex}^{t+1} are the speed of the evader in the X-axis direction at time t and time $t + 1$ respectively. v_{ey}^t and v_{ey}^{t+1} are the speed of the evader in the Y-axis direction at time t and time $t + 1$ respectively. In addition, the evader also has the ability to perceive the environment. When it senses the existence of obstacles around it, it will take the initiative to avoid obstacles. In order to avoid jitter, the evader will judge whether it is moving away from the obstacles according to the

perceived information. If so, it will maintain the current movement direction; otherwise, it will adopt the following strategies

$$\begin{cases} v_{\mathrm{ex}}^{t+1} = \begin{cases} \mathrm{random}\left(-v_{\mathrm{ex}}^{\max}, 0\right), v_{\mathrm{ex}}^{t} > 0 \\ \mathrm{random}\left(0, v_{\mathrm{ex}}^{\max}\right), \quad \text{otherwise} \end{cases} \\ v_{\mathrm{ey}}^{t+1} = \begin{cases} \mathrm{random}\left(-v_{\mathrm{ey}}^{\max}, 0\right), v_{\mathrm{ey}}^{t} > 0 \\ \mathrm{random}\left(0, v_{\mathrm{ey}}^{\max}\right), \quad \text{otherwise} \end{cases} \end{cases} \tag{9}$$

where, $v_{\mathrm{ex}}^{\max}$ and $v_{\mathrm{ey}}^{\max}$ are the maximum speed in the X-axis and Y-axis directions respectively. $\mathrm{random}\left(0, v_{\mathrm{ex}}^{\max}\right)$ indicates uniform sampling between 0 and $v_{\mathrm{ex}}^{\max}$, and the rest are the same.

3.3 Reward Function

Reward function is the key to guide the pursuer to learn effectively. Therefore, designing a reasonable reward function can improve the convergence speed and learning stability of the pursuer. Sparse reward is a simple reward function, which gives rewards only when the pursuer is in the termination state. If the pursuer completes the task, the environment gives a positive reward, which encourages the pursuer to continuously strengthen the action sequence. If the task fails, punishment is imposed to remind the pursuer to avoid some wrong behaviors. However, the sparse reward requires the pursuer to continuously explore the environment until the positive reward is obtained, which will reduce the learning efficiency and easily converge to the local optimal solution. Even for the complex environment, the pursuer cannot converge because it is difficult to explore the evader. Therefore, it is necessary to design a continuous reward function to guide the pursuer to approach the evader.

The reward function designed in this paper includes six parts: termination function, distance function, angle function, obstacle avoidance function, speed function and straight-line function. When the pursuer collides with an obstacle or exceeds the environmental boundary, the task fails; when the pursuer catches up with the evader and there is no collision, the task is successful. The mathematical expression of termination function is

$$r_{\mathrm{T}} = \begin{cases} -k_{\mathrm{T}}, \text{task failed} \\ k_{\mathrm{T}}, \quad \text{task success} \end{cases} \tag{10}$$

where, k_{T} is a positive real number, that is, the pursuer is rewarded when it succeeds and punished when it fails.

The mathematical expression of the distance function is

$$r_{\mathrm{D}} = k_{\mathrm{D}}(d_{\mathrm{l}} - d_{\mathrm{c}}) \tag{11}$$

where, d_{l} is the European distance between the pursuer and the evader at the last time, d_{c} is the European distance between the pursuer and the evader at the current time, and k_{D} is a positive real number. Equation 11 indicates that if the pursuer approaches the evader, it will receive a positive reward, otherwise it will be punished, and the reward intensity is directly proportional to the distance variation.

The mathematical expression of the angle function is

$$r_{\Theta} = \frac{k_{\Theta}}{\pi}\left(\frac{\pi}{2} - \alpha\right) \tag{12}$$

where, k_{Θ} is a positive real number. When $\alpha < \pi/2$, it means that the pursuer is approaching the evader, so as to obtain the reward given by the environment. On the contrary, when $\alpha > \pi/2$, it means that the pursuer is far away from the evader and will be punished.

The mathematical expression of obstacle avoidance function is

$$r = -k_{\mathrm{obs}} \sum_{i=1}^{9} \left(\frac{1}{d_i} - \frac{1}{r_{\mathrm{cap}}}\right) \tag{13}$$

$$r_{\mathrm{obs}} = \begin{cases} -r_{\mathrm{clip}}, r \le -r_{\mathrm{clip}} \\ r, \quad -r_{\mathrm{clip}} < r < 0 \end{cases} \tag{14}$$

where, r_{clip} is the minimum value of obstacle avoidance reward truncation, k_{obs} is a positive real number. The Eqs. 13 and 14 indicate that when the pursuer detects multiple obstacles within the perceived range or is close to the obstacles, the environment will impose punishment, and the punishment intensity is directly proportional to the degree of danger; when the pursuer does not perceive any obstacles, it will not be punished.

The mathematical expression of the straight-line function is

$$r_{\mathrm{line}} = k_{\Theta}\left(\frac{d_v - d_{\mathrm{thr}}}{d_{\mathrm{thr}}}\right) \tag{15}$$

where, d_v is the distance detected by the corresponding ray in the speed direction, as shown in d_5 in Fig. 3, and d_{thr} is the safe distance threshold. When $d_v < d_{\mathrm{thr}}$, it means that the pursuer is close to the obstacle and there is a collision risk, so the environment imposes punishment; when $d_v \ge d_{\mathrm{thr}}$, it indicates that there are no obstacles in the pursuer's forward direction, so as to obtain a positive reward to encourage the pursuer to move forward.

The mathematical expression of the speed function is

$$r_{\mathrm{V}} = \begin{cases} -k_{\mathrm{V}}\left(v_{\mathrm{c}} - v_{\mathrm{p}}\right), v_{\mathrm{c}} > v_{\mathrm{p}} \\ 0, \qquad\qquad \text{otherwise} \end{cases} \tag{16}$$

where, k_{V} is a positive real number, v_{c} is the critical speed. When the pursuer's speed is less than the critical speed, it will be punished, so as to encourage the pursuer to track quickly.

Combining the above six reward functions, the final reward function r is

$$r = r_{\mathrm{T}} + r_{\mathrm{D}} + r_{\Theta} + r_{\mathrm{obs}} + r_{\mathrm{line}} + r_{\mathrm{V}} \tag{17}$$

4 MPTD3 Algorithm

In TD3 algorithm, the experience data of the interaction between the pursuer and the environment will be placed in the experience pool, and in each round of iteration, the pursuer will randomly sample data from the experience pool to update the network. Therefore, the quality of the data in the experience pool determines the efficiency of neural network learning. When the pursuer falls into the local optimum, the relevant action sequence is continuously strengthened. At this time, the pursuer is unable to learn effective knowledge from the experience pool, so that it is difficult to escape from the local optimum. To solve this problem, this paper proposes Multiple Pools Twin Delay Deep Deterministic Policy Gradient (MPTD3) algorithm, which replaces a single experience pool with multiple experience pools to put the experience data separately, so as to improve the convergence speed of the algorithm.

4.1 TD3 Algorithm

TD3 algorithm is a deterministic deep reinforcement learning algorithm, which can effectively solve the problems of continuous state space and action space. The algorithm includes an actor network and two critic networks, and each network has a corresponding target network to ensure the stability of the training process.

In the critic network training stage, firstly, the target value and prediction value are calculated, then the critic loss is obtained, and finally the critic network is optimized by gradient descent method. The target value is calculated as follows

$$\begin{cases} y_j = r(s_j, a_j) + \gamma \min_{i=1,2} \widetilde{Q}\left(s_{j+1}, \widetilde{\pi}\left(s_{j+1}|\widetilde{\theta}\right) + \varepsilon|\widetilde{\omega}_i\right) \\ \varepsilon = N\left(0, \widetilde{\sigma}\right) \end{cases} \tag{18}$$

where, y_j is the target value, $r(s_j,\ a_j)$ is the immediate reward, γ is the discount coefficient, $\widetilde{\pi}\left(s_{j+1}, \widetilde{\theta}\right)$ is the target action in the state $s_{j\ +\ 1}$, $\widetilde{\theta}$ represent the parameters in the target actor network, $N(0, \widetilde{\sigma})$ is the smooth noise satisfying the normal distribution, $\widetilde{\sigma}$ is the variance of the smooth noise, $\widetilde{Q}\left(s_{j+1}, \widetilde{\pi}\left(s_{j+1}, \widetilde{\theta}\right) + N(0, \widetilde{\sigma})\ |\widetilde{\omega}_i\right)$ is the action value in the state $s_{j\ +\ 1}$, and $\widetilde{\omega}_i$ represent the parameters in the target critic network. When calculating the target value, the smaller output of the two target networks is selected to prevent the over estimation problem of the network. The critic loss is calculated as follows

$$L_{\mathrm{c}} = \frac{1}{B} \sum_{i=1}^{B} \sum_{j=1}^{2} \left(y_i - Q(s_i, a_i|w_j)\right)^2 \tag{19}$$

where, B is the total amount of data sampled during each update, $Q(s_i,\ a_i|w_j)$ is the action value in the state s_i, and w_j represent the parameters in the critic network.

In the actor network training stage, firstly, the predicted action in the state s_i is calculated, then the actor loss is obtained, and finally the actor network parameters are updated by gradient ascent method. The actor loss is calculated as follows

$$L_{\mathrm{a}} = \frac{1}{B} \sum_{i=1}^{B} Q(s_i, \pi(s_i|\theta)|w_1) \tag{20}$$

where, $\pi(s_i|\theta)$ is the predicted action in the state s_i,and θ represent the parameters of the actor network.

After updating the actor network and critic network, finally update all target networks by soft update, as shown in Eq. 21.

$$\begin{cases} \theta = \tau\theta + (1-\tau)\widetilde{\theta} \\ w_i = \tau w_i + (1-\tau)\widetilde{w}_i, i = 1, 2 \end{cases} \tag{21}$$

where, τ is the learning rate of soft update.

4.2 Multiple Experience Pool

As previously analyzed, when the pursuer falls into the local optimum, the learnability of data in a single experience pool is not strong for the pursuer. Replacing it with multiple experience pools can effectively ensure the quality of the data.

MPTD3 algorithm includes three experience pools: failure experience pool, success experience pool and temporary experience pool. The failure experience pool stores the failure experience data of the pursuer, and its capacity is set to M_{f}. The success experience pool stores the success experience data of the pursuer or the experience data that the pursuer can avoid obstacles for many times, and its capacity is set to M_{s}. The temporary experience pool stores the temporary experience data generated by the pursuer in the current generation, and its capacity is set to M_{t}. When the temporary experience pool overflows, if the pursuer does not reach the termination state, it is considered that its past decisions have a positive impact on the present, so the data overflowed from the temporary experience pool is stored in the success experience pool. The storage location of the remaining data in the temporary experience pool is determined by the final state. If successful, store it in the success experience pool, otherwise move it to the failure experience pool.

The data required for updating MPTD3 algorithm are sampled from the failure experience pool and the success

experience pool according to a certain proportion. In this way, even if the pursuer falls into the local optimum, it can still obtain the success experience data from the success experience pool to learn, so as to help the pursuer break away from the local optimum quickly. Assuming that the data sampling ratio of the pursuer is β, and the total amount of data sampled is B, so the amount of data sampled from the success experience pool and the failure experience pool are respectively

$$\begin{cases} b_1 = \begin{cases} D_s, D_s \le \beta B \\ \beta B, \text{otherwise} \end{cases} \\ b_2 = B - b_1 \end{cases} \tag{22}$$

where, b_1 is the amount of data sampled from the success experience pool, b_2 is the amount of data sampled from the failure experience pool, and D_s is the total amount of data in the success experience pool. The algorithm structure block diagram is shown in Fig. 4.

4.3 Network Structure

The network structure of MPTD3 algorithm is the same as TD3 algorithm. It contains two identical critic networks and an actor network. Each network has a corresponding target network. The input of the critic network includes the observation information obtained by the pursuer and the actions performed in this state. The corresponding dimensions are 20 and 2 respectively, and they are spliced into a 22 dimensional vector. Then it passes through two fully connected networks for feature extraction. The number of neurons in the first fully connected network is 400 and the second is 300. The activation function adopts Rectified Linear Unit (ReLU). Finally, the result of feature extraction is passed through a last fully connected network to obtain the corresponding state action value. The output dimension is 1 without activation function. The critic network structure is shown in Fig. 5a.

The input of the actor network is the observation information obtained by the pursuer. Similarly, it passes through two fully connected networks in turn, with the number of 400 and 300 neurons respectively, and both use the ReLU activation function. After feature extraction, the result is sent to the last fully connected network. Since the action includes speed and angle, the output dimension is 2. And in order to make the output of each action between [−1, 1], the activation function adopts Hyperbolic Tangent (Tanh). The actor network structure is shown in Fig. 5b.

In order to ensure that the updating process of actor network and critic network is more stable, the gradient value is trimmed [30], that is, when the gradient value is greater than the preset maximum gradient value, it is truncated to the maximum gradient value.

In this paper, MPTD3 algorithm is used to realize the pursuer's real-time obstacle avoidance and target tracking. When the pursuer moves in the environment, it can not only obtain the state information of itself and the evader, but also use its own sensor to sense partial environment around it. Then the obtained information is stacked together to form the observation information o. The observation information is sent to the pursuer's actor network to output actions a and guide the pursuer's movement. At the same time, the environment gives the pursuer immediate reward r, next observation o′ and state information *done*. Send the interactive experience data (o, a, r, o', *done*) into the temporary experience pool, and then judge the data flow direction in the temporary experience pool according to the current state of the pursuer. At each time step, the pursuer will randomly sample a batch of data from the success experience pool and failure experience pool according to the preset proportion for training. When the pursuer enters the termination state,

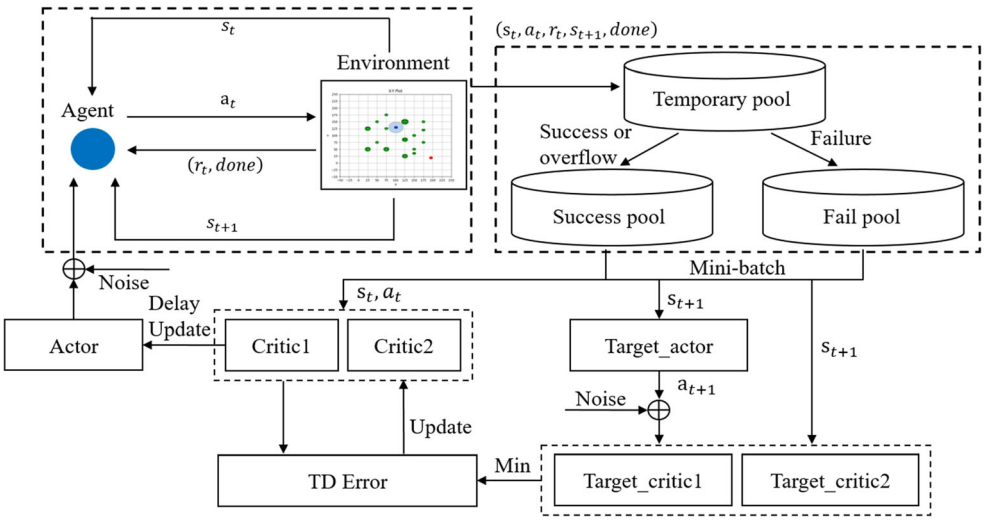


Fig. 4 Structure block diagram of MPTD3 algorithm

Algorithm 1 MPTD3

Initialize temporary experience pool P_t (maximum capacity is M_t)

Initialize the success experience pool P_s (maximum capacity is M_s)

Initialize the failure experience pool P_f (maximum capacity is M_f)

Initialize the critic network parameters $w_i \ (i=1,2)$ and the actor network parameters θ

Initialize target network parameters

$$\tilde{w}_i \leftarrow w_i \ (i=1,2), \ \tilde{\theta} \leftarrow \theta;$$

for $t=1$ to T **do**

1. Select action and add exploration noise

$$a \leftarrow \pi(o|\theta)+\varepsilon, \varepsilon = N(0,\sigma)$$

2. Pursuer executes action, then get the immediate reward r, next observation o' and state information *done*
3. Store the experience $(o,a,r,o',done)$ into the P_t. If the P_t overflows, the overflowed data will be sent to the P_s
4. Randomly sample experience $(o,a,r,o',done)$ with total amount of B from the P_s and P_f in proportion β
5. $\tilde{a} \leftarrow \tilde{\pi}(o'|\tilde{\theta})+\varepsilon, \varepsilon \leftarrow \text{clip}(N(0,\tilde{\sigma}),-c,c)$
6. $y = r+\gamma \min_{i=1,2} \tilde{Q}_i(o',\tilde{a}|\tilde{w}_i)$
7. Update w_i:

$$\arg\min_{w_i} B^{-1}\sum\left(y-Q_i(o,a|w_i)\right)^2$$

8. **if** t mod d **then**
9. Update θ:

$$\nabla_\theta J(\theta) = B^{-1}\sum \nabla_a Q_1(o,a|w_1)\big|_{a=\pi(o|\theta)} \nabla_\theta \pi(o|\theta)$$

10. Update target network:

$$\theta = \tau\theta+(1-\tau)\tilde{\theta}$$

$$w_i = \tau w_i+(1-\tau)\tilde{w}_i, i=1,2$$

11. **end if**
12. **if** *done* **then**
13. If the task is successful, put all the data in the P_t into the P_s at once; if the task fails, all the data in the P_t will be put into the P_f at one time
14. **end if**

end for

end this round and re-enter the next round of learning. The algorithm flow is shown in algorithm 1.

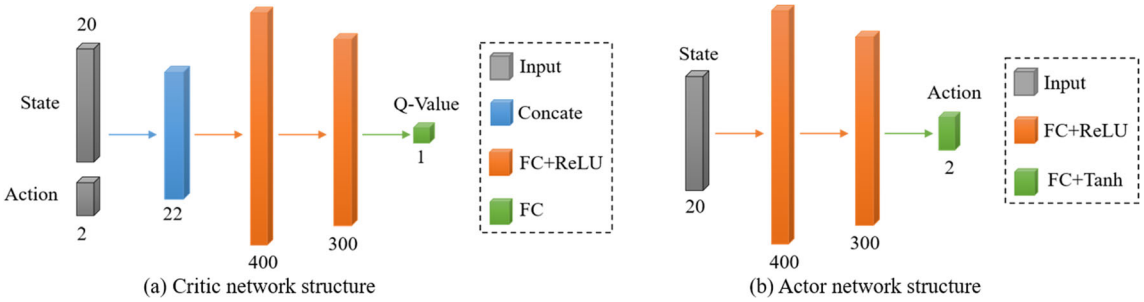
5 Experiments and Analysis

This paper constructs MPTD3 algorithm based on Pytorch1.6 and CUDA10.2 frameworks in Python environment. The optimizer used for updating parameters of critic network and actor network is Adma, and its learning rate *lr* is set to 0.0003. In order to ensure the convergence of the network, the maximum number of iterations N_e is set to 2000, and the maximum number of steps per iteration T is set to 500. The maximum capacity of the success experience pool and the failure experience pool is the same, and both are set to 1,000,000. If the pursuer does not enter the termination state within 200 steps, it is considered that the previous decision is valuable and the experience data needs to be saved to the success experience pool, so the maximum capacity of the temporary experience pool is set to 200. The maximum gradient value g_{clip} for gradient clipping is set to 0.5. All parameters and corresponding values are shown in Table 1.

5.1 MPTD3 Algorithm Verification

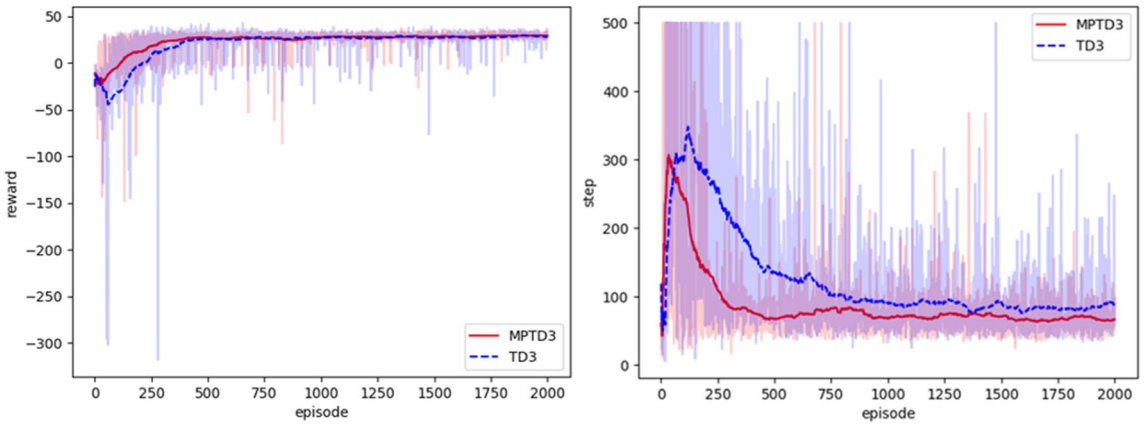
The experimental environment is a square area. In order to verify the pursuer's obstacle avoidance ability, there are several obstacles in the environment. The evader makes random movement and can avoid obstacles in the environment independently. The goal of the pursuer is to track the evader in real time and avoid obstacles autonomously. The current turn ends when the pursuer collides with an obstacle; crossing borders; maximum time step reached. When the environment is initialized, the locations of pursuer and evader will also be initialized randomly to enhance the randomness of the task. In this section, TD3 and MPTD3 algorithms are used for testing, and the test results are shown in Figs. 6, 7 and 8.

Figure 6a shows the cumulative reward curve of MPTD3 algorithm and TD3 algorithm in the training stage. It can be seen that both algorithms can enter the convergence state after 500 iterations, but the convergence speed of MPTD3 algorithm is slightly faster. Figure 6b shows the cumulative step curve. It can also be seen that MPTD3 algorithm has faster convergence speed. In addition, compared with TD3 algorithm, MPTD3 algorithm has fewer tracking steps, so the tracking path is shorter. Figure 7a, b show the tracking success rate curves in the training and testing stages respectively. It is obvious that the tracking success rate of MPTD3 algorithm is always higher than TD3 algorithm. Then the two trained algorithms are tested for 1000 rounds in the same environment. It is noted that their tracking success rate is more than 97%, and even MPTD3 algorithm is more than 99%. Figure 8 shows the tracking paths of the two algorithms in a simple



(a) Critic network structure (b) Actor network structure

Fig. 5 The network structure of critic and actor in MPTD3 algorithm. The input of the critic network is state and action, and the output is the corresponding state action value with 1 dimension. The input of the actor network is state and the output is action. The action is a two-dimensional tuple, including the changes of speed and angle



(a) Cumulative reward curve (b) Cumulative step curve

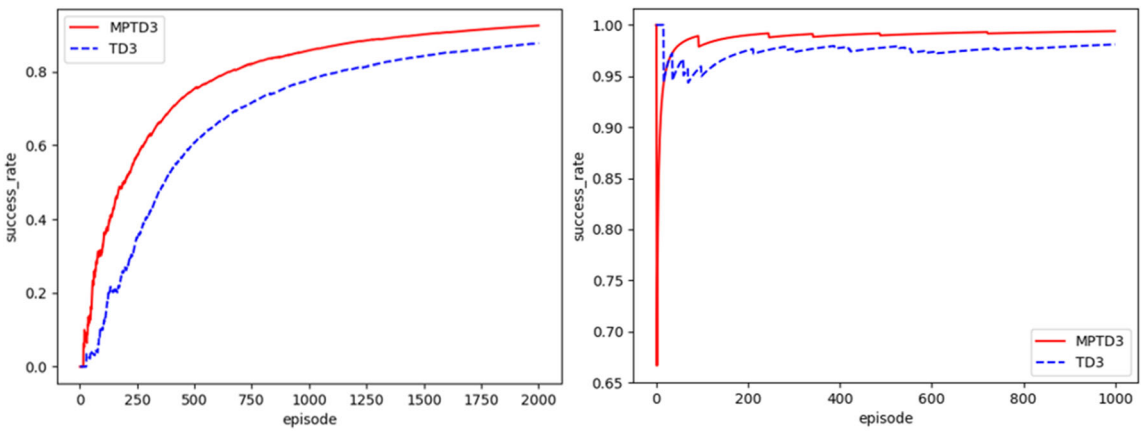
Fig. 6 Cumulative reward curve and cumulative step curve of training in simple environment without noise

environment, in which the red circle represents the pursuer, the blue circle represents the evader, the green circle represents the obstacle, and the cyan area around the blue circle represents the pursuer's perception range. It can be seen that both MPTD3 and TD3 algorithms can complete the task.

The experimental results show that MPTD3 algorithm has faster convergence speed and higher tracking success rate than TD3 algorithm.

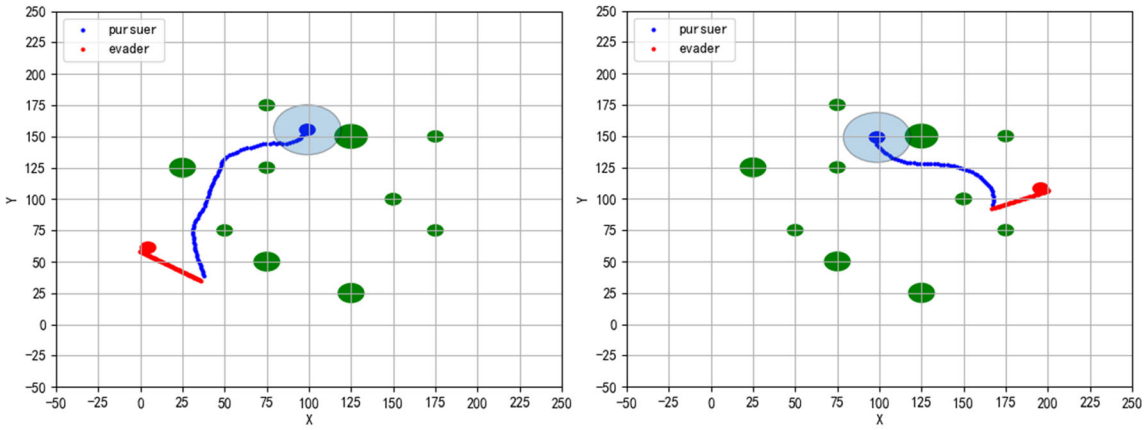
5.2 MPTD3 Robustness Verification

In practical application, there is often some noise interference in the perception information. In order to verify that MPTD3 has anti-interference ability, the noise that satisfy normal distribution is added to the perception information of UAV. The mean value of normal distribution is 0 and the variance is 1, so 99.73% of the noise data is within [−3, 3]. Since the perception



(a) Tracking success rate curve in training stage (b) Tracking success rate curve in testing stage

Fig. 7 Tracking success rate curve of training and testing in simple environment without noise



(a) TD3 algorithm tracking path (b) MPTD3 algorithm tracking path

Fig. 8 Tracking path in simple environment. The blue UAV represents the pursuer, the red UAV represents the evader, the green circle represents the obstacle, the cyan area represents the perception range of the pursuer, the blue track represents the motion path of the pursuer, and the red track represents the motion path of the evader

information has been normalized to [0, 1], the noise data is reduced by 20 times, so the noise proportion is less than 15%. The MPTD3 algorithm is trained again, and the test results are shown in Figs. 9 and 10.

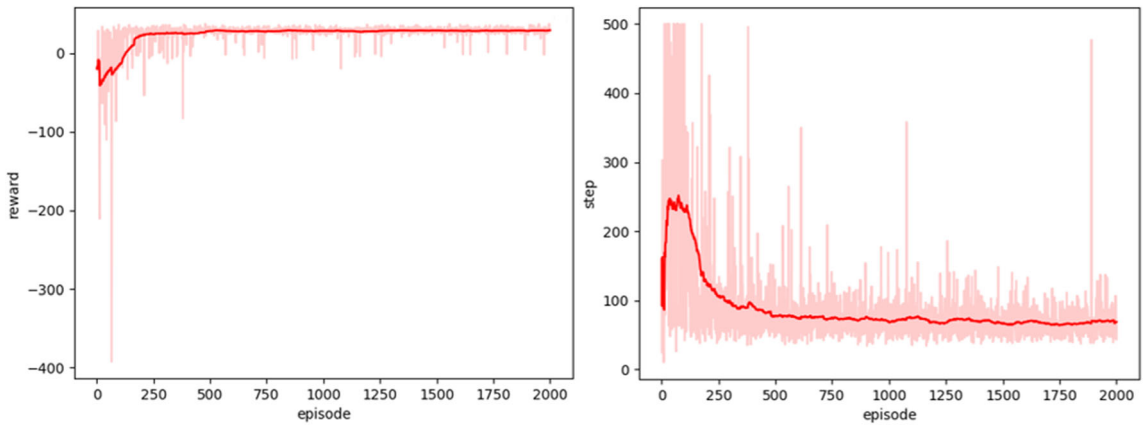
Figure 9a is the cumulative reward curve during training, and Fig. 9b is the cumulative step curve. Combined with these two curves, it can be seen that the MPTD3 algorithm tends to converge when the iteration is about 250 rounds. Compared with the training results in Section 5.1, it is found that adding noise to the perception information improves the convergence speed of the algorithm. The reason is that adding noise enriches the diversity of data and reduces the correlation between data. Figure 10a is the tracking success rate curve during training, and Fig. 10b is the tracking success rate curve during testing. Compared with the experimental results in Section 5.1, it can also be seen that the tracking success rate is improved after adding noise, even 99.8% in the test. Through analysis, due to the noise in the perception information, the pursuer will keep a long distance from the obstacle to avoid collision, which is safer and makes the tracking success rate higher.

The experimental results show that MPTD3 algorithm has good anti-interference ability.

5.3 MPTD3 Generalization Verification

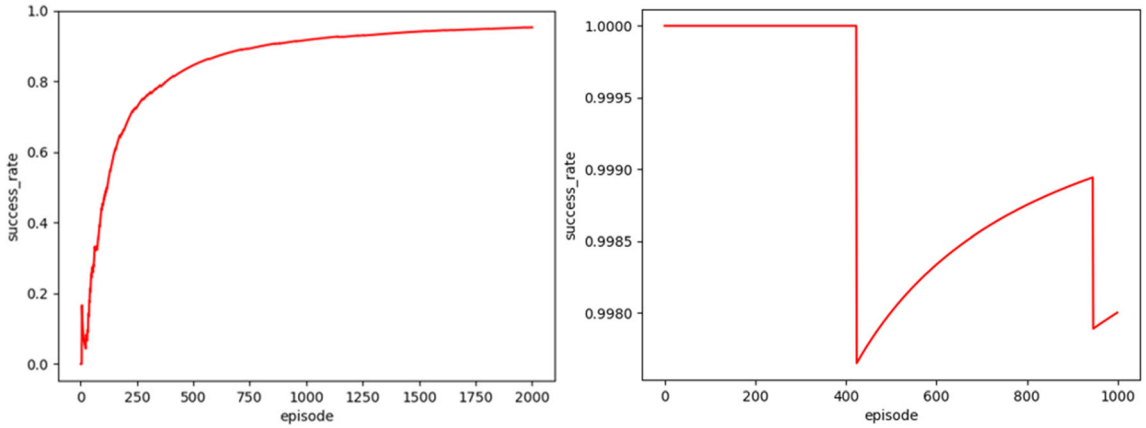
In order to verify that the trained pursuer has the ability of environmental generalization, this section applies the previously trained pursuer to test in a new environment, where obstacles are denser, obstacle avoidance and target tracking are more difficult. Also test 1000 rounds, and the test results are shown in Figs. 11 and 12.

Figure 11a shows the tracking success rate of the two algorithms after 1000 rounds of testing in a complex environment, and there is no noise interference in the pursuer's perception information. The tracking success rates of both MPTD3 and TD3 algorithm are above 94% and still maintained at a high level, especially the tracking success rate of MPTD3 algorithm is more than 98%. Figure 12 shows the tracking paths of the two algorithms in a complex environment. It can be seen that both MPTD3 and TD3 algorithms can complete tasks. In



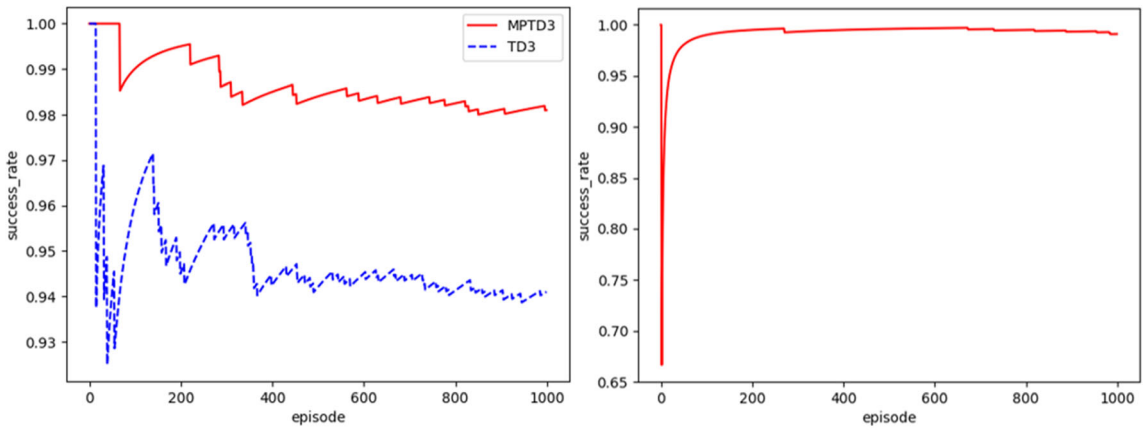
(a) Cumulative reward curve (b) Cumulative step curve

Fig. 9 Cumulative reward curve and cumulative step curve of training in simple environment with noise



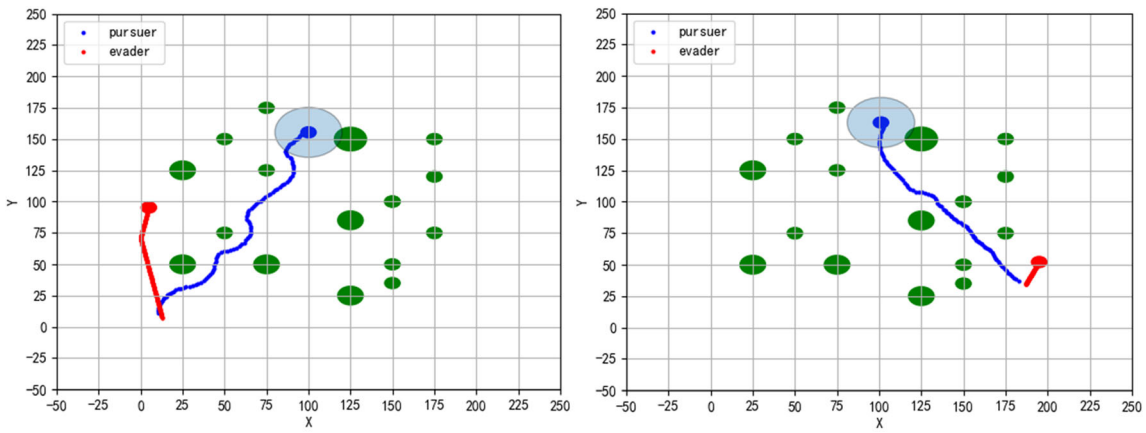
(a) Tracking success rate curve in training stage (b) Tracking success rate curve in testing stage

Fig. 10 Tracking success rate curve of training and testing in simple environment with noise



(a) Without noise (b) With noise

Fig. 11 Tracking success rate curve of testing in complex environment. There is no noise in the pursuer's perception information in (**a**), but there is noise in (**b**)



(a) TD3 algorithm tracking path (b) MPTD3 algorithm tracking path

Fig. 12 Tracking path in complex environment. There is no noise in the pursuer's perception information

Table 1 MPTD3 algorithm parameters setting table

Parameter	Set value	Parameter	Set value
lr	0.0003	τ	0.005
B	256	β	0.85
N_e	2000	T	500
γ	0.99	σ	0.1
d	2	$\widetilde{\sigma}$	0.2
c	0.5	M_s	1,000,000
M_f	1,000,000	M_t	200
r_{cap}	15	v_p^{max}	2
Δv_{max}	0.5	$\Delta\theta_{max}$	$\pi/6$
v_{ex}^{max}	1	v_{ey}^{max}	1
v_e^{max}	$\sqrt{2}$	h	300
w	300	k_T	10
k_D	0.05	k_{obs}	1.5
k_Θ	0.2	d_{thr}	10
r_{clip}	0.5	v_c	0.5
k_V	0.2	g_{clip}	0.5

addition, we also tested the trained pursuer with noise interference in the perception information for 1000 rounds in this complex environment. The tracking success rate curve is shown in Fig. 11b, which also reaches more than 98%.

The experimental results in complex environment show that both TD3 and MPTD3 algorithms have good environment generalization ability, and the effect of MPTD3 algorithm is better. Therefore, when training the algorithm, the pursuer can interact in a simple environment to reduce the training difficulty and shorten the training time.

6 Conclusion

Aiming at the problem of autonomous obstacle avoidance and target tracking of UAV in complex environment, an improved deep reinforcement learning algorithm MPTD3 is proposed in this paper. The experimental results show that the pursuer with environmental perception can adapt to more complex environment by using MPTD3 and TD3 algorithms, and both show excellent environmental generalization ability. However, compared with TD3 algorithm, MPTD3 algorithm has better convergence performance and higher tracking success rate due to the use of multiple experience pools mechanism and gradient clipping. In addition, MPTD3 algorithm has good robustness and can effectively overcome the noise interference in practical application.

There are still some limitations. On the one hand, the motion mode of real UAV is three-dimensional motion, which is modeled as two-dimensional translation in this paper; on the other hand, the proposed algorithm has not been tested in the actual environment, which has complex obstacles. The future work will mainly focus on building the three-dimensional model of UAV, training agents to realize autonomous obstacle avoidance and target tracking in the three-dimensional environment, and then further expand to the actual environment.

Acknowledgements This work was supported by the National Natural Science Foundation of China under the grant number 61903133 and 61733004.

Code or Data Availability Source code in Python generated during the current study is available from the corresponding author on reasonable request.

Author Contributions Guoqiang Xu contributed to the design, implementation and manuscript writing of the research; Weilai Jiang and Yaonan Wang contributed to the guidance of the experiment and the revision of the manuscript.

Funding This research was funded by National Science Foundation of China under the grant for Prof. Yaonan Wang having grant number 61733004 and Prof. Weilai Jiang having grant number 61903133.

Declarations

Ethical Approval No applicable as this study does not contain biological applications.

Consent to Participate All authors of this research paper have consented to participate in the research study.

Consent to Publication All authors of this research paper have read and approved the final version submitted.

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Shirani, B., Najafi, M., Izadi, I.: Cooperative load transportation using multiple UAVs. Proc. Aerosp. Sci. Technol. **84**, 158–169 (2019). https://doi.org/10.1016/j.ast.2018.10.027
2. Khan, M.A., Cheema, T.A., Ullah, I., Noor, F., Aziz, M.A.: A dual-mode medium access control mechanism for UAV-enabled intelligent transportation system. Proc. Mob. Inf. Syst. (2021). https://doi.org/10.1155/2021/5578490
3. Sung, I., Nielsen, P.: Zoning a service area of unmanned aerial vehicles for package delivery services. Proc. J. Intel. Robot. Syst. **97**, 719–731 (2020)
4. Umemoto, K., Endo, T., Matsuno, F.: Dynamic cooperative transportation control using friction forces of n multi-rotor unmanned aerial vehicles. Proc. J. Intell. Robot. Syst. **100**, 1085–1095 (2020). https://doi.org/10.1007/s10846-020-01212-1
5. Liu, X., Ansari, N.: Resource allocation in UAV-assisted M2M communications for disaster rescue. Proc. IEEE Wirel. Commun. Lett. **8**(2), 580–583 (2018)
6. Wang, Y., Su, Z., Xu, Q., Li, R., Luan, T.H.: Lifesaving with Rescuechain: Energy-Efficient and Partition-Tolerant Blockchain Based Secure Information Sharing for UAV-Aided Disaster Rescue. In: Proceeding of IEEE Conference on Computer Communications (2021)

7. Dong, J., Ota, K., Dong, M.: UAV-Based Real-Time Survivor Detection System in Post-Disaster Search and Rescue Operations. In: Proceeding of IEEE Journal on Miniaturization for Air and Space Systems (2021)
8. Stampa, M., Sutorma, A., Jahn, U., Thiem, J., Wolff, C., Röhrig, C.: Maturity levels of public safety applications using unmanned aerial systems: a review. Proc. J. Intel. Robot. Syst. **103**(1), 1–15 (2021). https://doi.org/10.1007/s10846-021-01462-7
9. Lyu, J., Zeng, Y., Zhang, R., Lim, T.J.: Placement optimization of UAV-mounted mobile base stations. Proc. IEEE Commun. Lett. **21**(3), 604–607 (2016)
10. Wu, Y., Yang, W., Guan, X., Wu, Q.: UAV-Enabled Relay Communication under Malicious Jamming: Joint Trajectory and Transmit Power Optimization. Proc. IEEE Trans. Veh. Technol. (2021)
11. Cetin, O., Zagli, I., Yilmaz, G.: Establishing obstacle and collision free communication relay for UAVs with artificial potential fields. Proc. J. Intel. Robot. Syst. **69**(1), 361–372 (2013). https://doi.org/10.1007/s10846-012-9761-y
12. Oh, D., Lim, J., Lee, J.K., Baek, H.: Airborne-relay-based algorithm for locating crashed UAVs in GPS-denied environments. In: Proceeding of 2019 IEEE 10th annual ubiquitous computing, Electronics & Mobile Communication Conference (UEMCON). IEEE (2019)
13. Huang, Z., Zhang, T., Liu, P., Lu, X.: Outdoor independent charging platform system for power patrol UAV. In: Proceeding of 2020 12th IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), pp. 1–5. IEEE (2020)
14. Chang, A., Jiang, M., Nan, J., Zhou, W., Li, X., Wang, J., He, X.: Research on the application of computer track planning algorithm in UAV power line patrol system. In: Proceeding of Conference Series (Vol. 1915, No. 3, p. 032030). IOP Publishing (2021)
15. Pham, H.X., La, H.M., Feil-Seifer, D., Nguyen, L.V.: Autonomous UAV navigation using reinforcement learning. arXiv 2018. arXiv preprint arXiv:1801.05086 (2018)
16. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature. **518**(7540), 529–533 (2015)
17. Yan, C., Xiang, X., Wang, C.: Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. Proc. J. Intel. Robot. Syst. **98**, 297–309 (2020). https://doi.org/10.1007/s10846-019-01073-3
18. Yao, Q., Zheng, Z., Qi, L., Yuan, H., Guo, X., Zhao, M., Liu, Z., Yang, T.: Path planning method with improved artificial potential field—a reinforcement learning perspective. Proc. IEEE Access. **8**, 135513–135523 (2020)
19. Hausknecht, M., Stone, P.: Deep Recurrent Q-Learning for Partially Observable MDPs. In: Proceeding of AAAI Fall Symposium Series (2015)
20. Singla, A., Padakandla, S., Bhatnagar, S.: Memory-Based Deep Reinforcement Learning for Obstacle Avoidance in UAV with Limited Environment Knowledge. Proc. IEEE Trans. Intell. Transp. Syst. (2019)
21. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
22. Rodriguez-Ramos, A., Sampedro, C., Bavle, H., De La Puente, P., Campoy, P.: A deep reinforcement learning strategy for UAV autonomous landing on a moving platform. Proc J Intel Robot Syst. **93**(1–2), 351–366 (2019). https://doi.org/10.1007/s10846-018-0891-8
23. Li, B., Yang, Z.P., Chen, D.Q., Liang, S.Y., Ma, H.: Maneuvering target tracking of UAV based on MN-DDPG and transfer learning. Proc. Defence Technol. **17**(2), 457–466 (2021). https://doi.org/10.1016/j.dt.2020.11.014
24. Wan, K., Gao, X., Hu, Z., Wu, G.: Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning. Proc. Remote Sens. **12**(4), 640 (2020)
25. Sampedro, C., Rodriguez-Ramos, A., Bavle, H., Carrio, A., de la Puente, P., Campoy, P.: A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. Proc. J. Intel. Robot. Syst. **95**(2), 601–627 (2019). https://doi.org/10.1007/s10846-018-0898-1
26. Wang, C., Wang, J., Shen, Y., Zhang, X.: Autonomous navigation of UAVs in large-scale complex environments: a deep reinforcement learning approach. Proc. IEEE Trans. Veh. Technol. **68**(3), 2124–2136 (2019)
27. Song, D.R., Yang, C., McGreavy, C., Li, Z.: Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge. In: Proceeding of 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 311–318. IEEE (2018)
28. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Proceeding of International Conference on Machine Learning, pp. 1587–1596. PMLR (2018)
29. Li, B., Gan, Z., Chen, D., Sergey Aleksandrovich, D.: UAV maneuvering target tracking in uncertain environments based on deep reinforcement learning and meta-learning. Proc. Remote Sens. **12**(22), 3789 (2020). https://doi.org/10.3390/rs12223789
30. Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of MAPPO in cooperative multi-agent games. arXiv preprint arXiv:2103.01955 (2021)

Guoqiang Xu is a graduate student in the College of Electrical and Information Engineering at Hunan University, Changsha, China. He received the B.S. degree in the College of Engineering from Anhui Agricultural University, Hefei, China, in 2020. His research interests include the application of deep reinforcement learning in UAV obstacle avoidance and target tracking

Weilai Jiang is an associate professor in the College of Electrical and Information Engineering at Hunan University. He received the B.S. degree from Beijing Institute of Technology in 2010, and Ph.D. degrees from Beihang University in 2016. At present, he is mainly engaged in the research of UAV control, deep reinforcement learning and autonomous unmanned system.

Zhaolei Wang is a senior engineer of Beijing Aerospace Automatic Control Institute – Science and Technology on Aerospace Intelligent Control Laboratory. He received the B.S. degree from Beijing Institute of Technology in 2009, and Ph.D. degrees from Beihang University in 2015. At present, his main research interests include flight control, intelligent control, fault diagnosis and reinforcement learning, etc.

Yaonan Wang received his Ph.D. from Hunan University in 1995. He is now an academician of Chinese engineering, an expert in robot technology and intelligent control, a professor and doctoral supervisor of Hunan University, and the director of the National Engineering Laboratory of Robot Vision and Control Technology. His research interests are intelligent control theory and application, machine learning and image recognition, intelligent robot technology and application, autonomous UAV system technology, intelligent manufacturing measurement and control technology, etc.